

TOSHIBA

TLCS-9000/16

INSTRUCTION SET MANUAL

Version 2.2

10 Feb 1994

TOSHIBA CORPORATION

Legend

B	Byte (8 bits)
W	Word (16 bits)
D	Double word (32 bits)
SS	Operation size (0: word, 1: double word)
S1·S0	Operation size (01: byte, 10: word, 11: double word)
#	Immediate data
#4	4-bit immediate data
#8	8-bit immediate data
#16	16-bit immediate data
#32	32-bit immediate data
dst	Destination for data transfer or operation result store Also, depending on the instruction, may mean source for operation data. When more than one, expressed as dst1, dst2....
src	Source for transfer data or operation data. When more than one, expressed as src1, src2....
num	Source for numerical data. When more than one, expressed as num1, num2....
abs	Absolute operand address. Bit size may be denoted explicitly as abs13.
disp	Displacement. Bit size may be denoted explicitly as disp9 or disp13.
Z,S,V,C	Zero flag, Sign flag, Overflow flag, Carry flag
@	Operation size. B: byte, W: word, D: double word
Reg	General-purpose registers R0 to R15
SEA	Short addressing mode
LEA	Long addressing mode
MEM	Effective address value

R<3:0>	General-purpose register
T<3:0>	General-purpose register
M<7:0>	Long addressing mode
N<7:0>	Long addressing mode
m<5:0>	Short addressing mode
D<12:1>	Displacement value. Sign-extended for use.
D<8:1>	Displacement value. Sign-extended for use.
A<12:0>	Absolute address. Sign-extended for use.
DD	Indicates store direction. - In G format, 0: register indicated by R<3:0>, 1: address specified by m<5:0>. - In A format, 0: address indicated by :A<12:0>, 1: address specified by m<5:0>.
II	Indicates the meaning of source field (M<7:0>) in G format. 0: immediate data (8 bits, signed) 1: data specified by long addressing mode
B<2:0>	Bit number
S<4:0>	Offset value of bit to be operated on. (Used in bit field instruction)
W<3:0>	Width of bit to be operated on. (Used in bit field instruction)
C<3:0>	Type of condition to be tested for in conditional jump or conditional loop (jump non zero) instruction.
V<3:0>	Vector value (Used in software interrupt instruction)

When the operation size is double word, odd numbered registers cannot be specified. RBn, RWn, and RDn are byte, word, and double word general-purpose registers respectively.

Flag Changes

0	Reset to 0.
1	Set to 1.
-	No change.
*	Changes according to operation result.
?	Indeterminate value.

When flags change according to the result of an operation, they generally follow the rules described below. Exceptions to the rules are described later for each instruction.

Z	Zero flag. Set to 1 when the operation results in zero; otherwise, set to 0.
S	Sign flag. The most significant bit (MSB) of the operation result is copied.
V	Overflow flag. Set to 1 when an overflow occurs as a result of the operation; otherwise, set to 0.
C	Carry flag. Set to 1 when a carry or borrow from the MSB occurs as a result of the operation; otherwise, set to 0.

When a flag register is the target of an operation, the register is updated according to the result of the operation. In this case, updating using the operation result takes precedence over the above rule.

(Example) ADD.B CC,1 Adds 1 to the value in the CC register (flag). The result (CC register value + 1) is set in the flag register.

Instruction formats

Most instructions have multiple instruction formats for enabling various combinations of operands. Addressing modes for operands to be specified and combinations of modes differ depending on the instruction format. Instruction length and instruction execution time may differ, too.

The programmer need not specify instruction formats because the assembler automatically selects appropriate formats.

However, if desired, the programmer may specify a specific format. In such cases, add a colon and the name of a format, for example, : G to the end of the instruction mnemonic.

S format	Abbreviated type The types of operands which can be specified are restricted. However, this format is useful for minimizing the number of execution clocks and maximizing the efficiency of the object code.
G format	General binomial operation type General format for instructions which need two operands. Operands can be specified freely to provide powerful instruction functions.
I format	Immediate type Useful for speeding up execution and maximizing the efficiency of object code for frequently used instructions which need an immediate data for source operands. If the operation size is double word, immediate data are allocated in the order, lower word followed by upper word.
A format	Absolute binomial operation type Useful for speeding up execution and maximizing the efficiency of object code for instructions which need two operands, one of which is an absolute address.

(Examples) ADD.D RD6,4

Automatically generates a 1-word instruction code in S format.

ADD.D:G RD6,4

Unconditionally generates a 2-word instruction code in G format.

ADD.D:I RD6,4

Unconditionally generates a 3-word instruction code in I format.

ADD.W (RD2),RW7

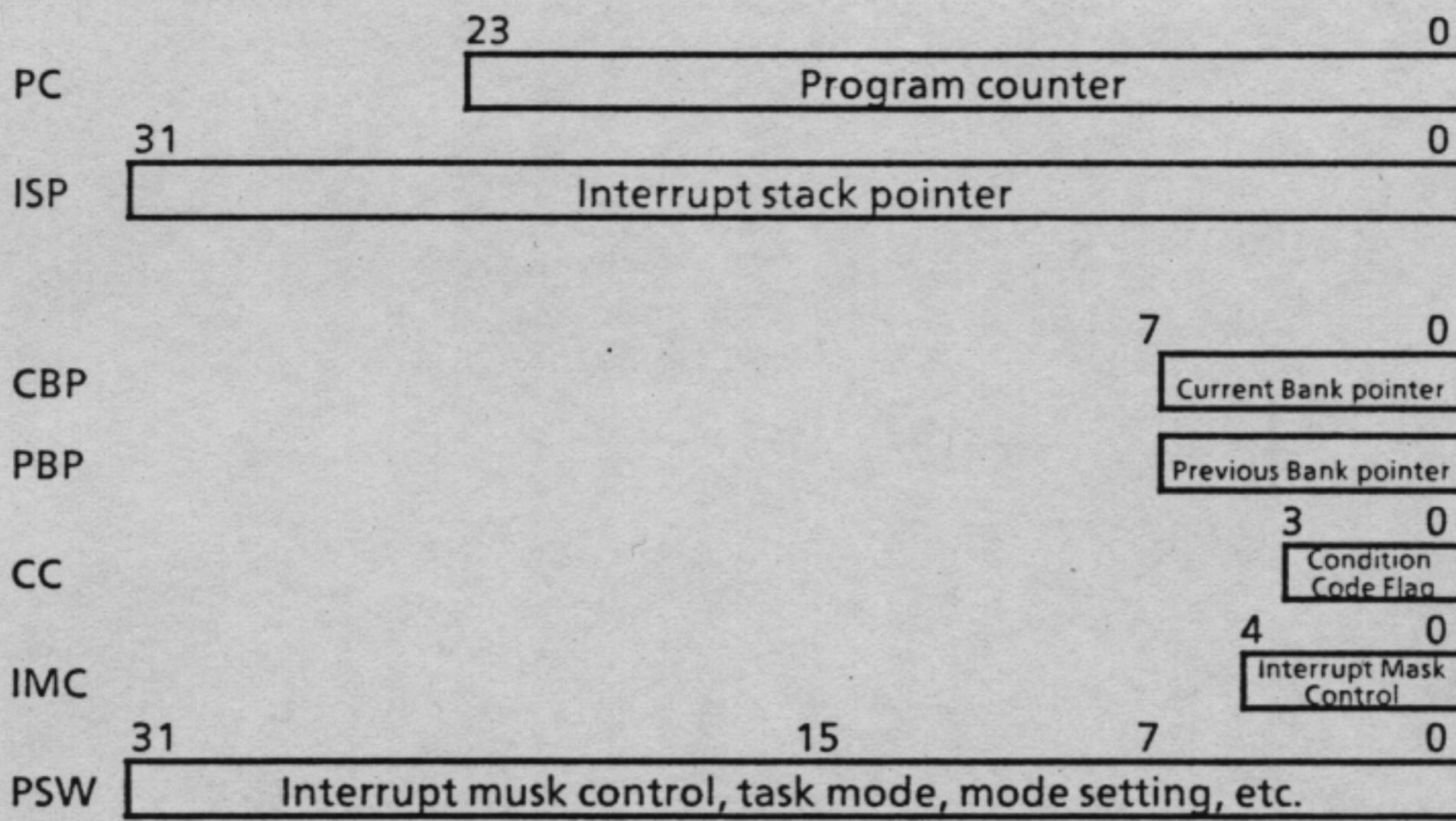
Automatically generates an instruction code in G format.

ADD.W:S (RD2),RW7

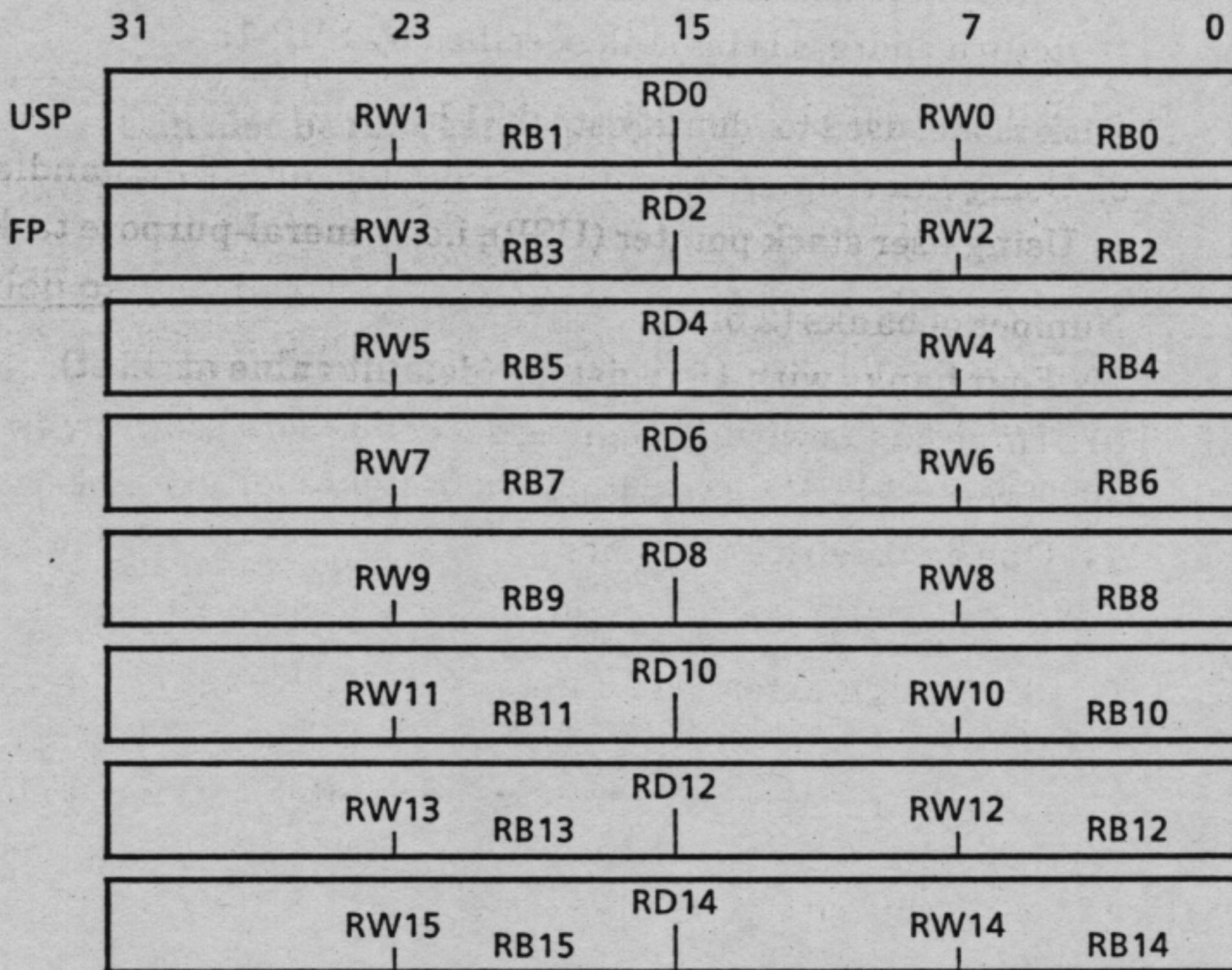
Assemble error occurs due to inhibited addressing mode combination in S format.

Register models

Special register (Non-bank section)



General-purpose register (Bank section)



(Note) RW0 or RD0 can be used as a user stack pointer (USP) only when task mode (TM) in PSW = 1. Otherwise, handled as general-purpose registers.

- PSW bits

PSW	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RA	-	TM	-	BS	-	-	-	-	-	-	-	-	-	-	-
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	-	-	-	-	SP	SS	-	EI		IM		Z	S	V	C	

“-” indicates an undefined (reserved) bit; 0 is read and data written in the bit is ignored.

All bits are cleared to “0” by resetting.

Bit 8 to 4 are the same as the IMC register referenced.

Bit 3 to 0 are the same as the CC register referenced.

RA Return address stack (1 bit)

0: Return address is in memory stack.

1: Return address is in bank specified by PBP-1.

TM Task mode, used to identify stack (1 bit)

0: Using interrupt stack pointer (ISP), i.e., interrupt handler

1: Using user stack pointer (USP), i.e., general-purpose task

BS Number of banks (2 bits)

00: Four banks with 16 registers (default value at reset)

01: Three banks with 12 registers

10: Two banks with 8 registers

11: One bank with 4 registers

SP User stack pointer size (1 bit)

0: 16 bits (sign-extended for use)

1: 32 bits

Can also be referenced as user stack pointer, interrupt stack pointer and frame pointer size.

SS Single step (1 bit)

0: Normal operation

1: Requests an interrupt for single step.

EI Enable interrupt (1 bit)
0: Disables maskable interrupts.
1: Enables maskable interrupt according to mask level.
Can also be referenced as the interrupt mask control register (IMC) as well as processor status word (PSW).
Automatically cleared at interrupt entry.

IM maskable interrupt mask (4 bits)
0000: Enables all maskable interrupts.
1111: Disables all maskable interrupts.
Can also be referenced as the IMC as well as PSW.

Z Zero (set when the operation result is all 0s.)
Can also be referenced as the condition code register (CC) as well as PSW.

S Sign bit (the most significant bit of the operation result is copied.)
Can also be referenced as the CC as well as PSW.

V Overflow viewed as signed binary numeral.
Can also be referenced as the CC as well as PSW.

C Carry or borrow.
Can also be referenced as the CC as well as PSW.

Brief Description of Registers

◎ PC Program counter.

Always allocated at an even numbered address, since instructions come in units of words. Thus, the least significant bit of the PC is 0.

◎ USP ... User stack pointer.

RW0 and RD0 is used as USP when task mode is 1; otherwise, handled as a general-purpose register.

When USP consists of 16 bits, RW0 is used as USP; when 32 bits, RD0 (RW0 + RW1) is used.

When the stack pointer consists of 16 bits, the unused upper bits are sign-extended for reading. Writing data to these bits are not accepted. USP is undefined by resetting.

◎ ISP Interrupt stack pointer

Used as ISP when task mode is 0.

When the stack pointer consists of 16 bits, bits 0 to 15 are used as ISP; when 32 bits, bits 0 to 31 are used. When the stack pointer consists of 16 bits, the unused upper bits are sign-extended for reading. Writing data to these bits are not accepted. ISP is undefined by resetting.

◎ CBP ... Current bank pointer

The current register bank number is set in this register. Writing to this register changes banks.

The unused upper bits are zero-extended to be read. Writing to these bits are invalid.

◎ PBP ... Previous bank pointer

The bank number of the previously used bank is set in this register when banks are changed by an interrupt. This is because the PC and processor status word (PSW) of the program interrupted are saved in the register bank area. Thus PBP indicates the bank position which holds the information for the return.

The unused upper bits are zero-extended to be read. Writing to these bits are invalid. PBP is undefined by resetting.

◎ PSW ... Processor status word

Data including the status of the processor such as interrupt mask control, task mode, and mode setting are set in this register.

Undeterminate (reserved) bits are set to "0" when reading. Writing to these bits are invalid. All bits are cleared to "0" by resetting.

◎ CC Condition code register

Data the same as the CC bit in the PSW are set in this register.

Used only when accessing flags indicating the operation result in the PSW.

The unused upper bits are zero-extended to be read. Writing to these bits are invalid.

◎ IMC ... Interrupt mask control register

Data the same as the IMC bit in the PSW are set in this register.

Used only when accessing bits related to interrupt mask control in the PSW.

The unused upper bits are zero-extended to be read. Writing to these bits are invalid.

◎ FP Frame pointer

Used when creating or deleting a stack frame. When the stack pointer consists of 16 bits, RW2 is used as FP; when 32 bits, RD2 (combined RW2 and RW3) is used. FP is undefined by resetting.

◎ General purpose register

Undefined by resetting.

Data Organization TLCS-9000 core CPU data organization

1. Bit organization of data

The TLCS-9000 core CPU can handle 1-, 8- (byte), 16- (word), and 32-bit (double word) data in most instructions. It can also handle 4-bit data in BCD operation instructions or data of any size (up to 16 bits) in bit field instructions.

Unsigned binary data are expressed as: bit number $n = 2^n$. For example, byte data are expressed as follows:

$$2^7 \cdot B_7 + 2^6 \cdot B_6 + 2^5 \cdot B_5 + 2^4 \cdot B_4 + 2^3 \cdot B_3 + 2^2 \cdot B_2 + 2^1 \cdot B_1 + 2^0 \cdot B_0$$

Bit number	MSB	LSB
7 6 5 4 3 2 1 0		
Binary code	1 0 1 0 0 1 1 1	
Decimal	167	
Hexadecimal	A7H	

Signed binary data uses the MSB as a sign bit. When MSB = 0, the sign is positive; when MSB = 1, it is negative.

For example, byte data are expressed as follows:

$$-2^7 \cdot B_7 + 2^6 \cdot B_6 + 2^5 \cdot B_5 + 2^4 \cdot B_4 + 2^3 \cdot B_3 + 2^2 \cdot B_2 + 2^1 \cdot B_1 + 2^0 \cdot B_0$$

Bit number	MSB	LSB
7 6 5 4 3 2 1 0		
Binary code	1 0 1 0 0 1 1 1	
Decimal	-89	

2. Data organization in general-purpose registers

There are up to 16 byte registers, indicated by RB0 to RB15. They use the lower 8 bits of a word register.

Bit number	MSB	LSB
7 6 5 4 3 2 1 0		

There are up to 16 word registers, indicated by RW0 to RW15.

Each word register contains the byte register as its lower 8 bits and can be used as the upper half or the lower half of the double word register.

Bit number	MSB	LSB
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		

There are up to 8 double word registers, indicated by RD0 to RD14 (even numbers only). RD_n uses RW_n as the lower word and RW_{n+1} as the upper word.

Bit number	MSB	LSB
31 30 29		
	5 4 3 2 1 0	

3. Data organization in memory

With the TLCS-9000 core CPU, memory is accessed in two ways: by instruction fetch or data access. Restrictions on data organization in memory differ depending on the access method.

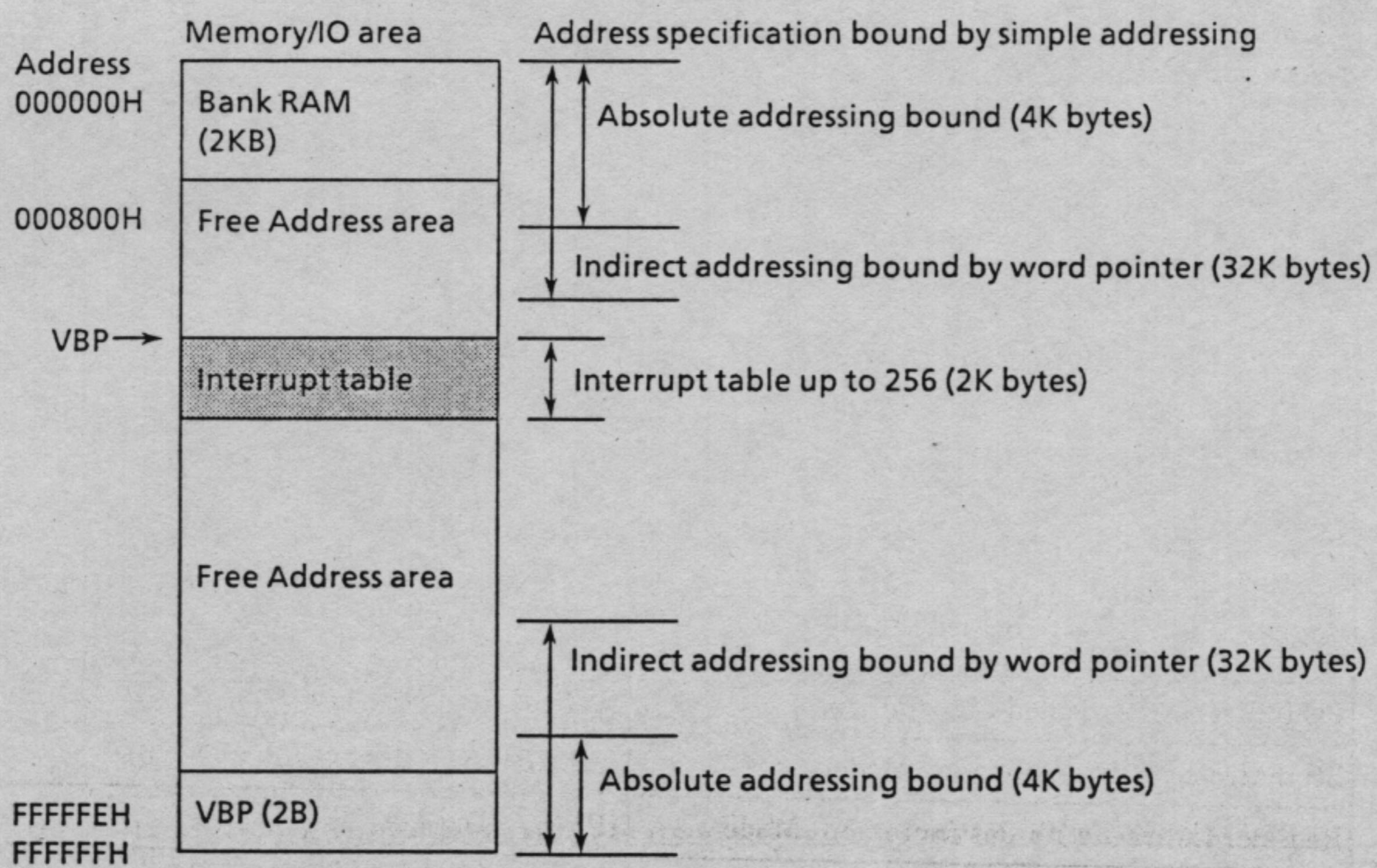
In an instruction fetch, the start address can only be accessed from an even numbered address, in units of words. Whereas, in a data access, the start address can be accessed from either an even or an odd numbered address, in units of bytes, words, or double words. From the point of memory access execution time, it is better to organize word or double word data starting from an even numbered address.

An example of data organization in memory is shown below.

		Address 0	Data								
			MSB	LSB							
			7	6	5	4	3	2	1	0	(Bit number)
Instruction code	Even numbered address		□	□	□	□	□	□	□	□	
	Odd numbered address		□	□	□	□	□	□	□	□	
Byte data		Address n	□	□	□	□	□	□	□	□	
Word data		Address n	□	□	□	□	□	□	□	□	(Lower byte)
		Address n + 1	□	□	□	□	□	□	□	□	(Upper byte)
Double word data		Address n	□	□	□	□	□	□	□	□	(Byte on LSB side)
		Address n + 1	□	□	□	□	□	□	□	□	
		Address n + 2	□	□	□	□	□	□	□	□	
		Address n + 3	□	□	□	□	□	□	□	□	(Byte on MSB side)

Memory map

16M-byte Memory/IO area (specified by 24 bits)



1. **Vector base pointer (VBP):** Initial program counter value after CPU reset is set. Variable from 000000H in units of 2K bytes.
2. An interrupt is allocated in the interrupt Table starting from an address specified by the VBP in units of 8 bytes.

Addressing modes

TLCS-9000 Series has 14 Addressing Modes. These Addressing modes use one of the two addressing mode format, Short Addressing Mode (SEA) and Long Addressing Mode (LEA).

Effective Addressing Mode Categories.

General Register Direct Mode	RBn / RWn / RDn
Special Register Direct Mode	SP / ISP / PBP / CBP / PSW / IMC / CC
Register Indirect Mode	(RWn / RDn)
Register Indirect with Displacement Mode	(RWn / RDn + disp 9 / 13 / 20 / 24)
SP Indirect with Displacement Mode	(SP + disp 7 / 9 / 20)
PC Indirect with Displacement Mode	(PC + disp 9 / 20)
Bank Register Indirect with Displacement Mode	(RWn / RDn × 8 + disp 9 / 20)
Register Indirect with Scaled Index Mode	(RWm / RDm + RWn / RDn × ss + disp 9 / 20) (RWn / RDn × ss + disp 9/20)
PC Indirect with Scaled Index Mode	(PC + RWn / RDn × ss + disp 9 / 20)
SP Indirect with Scaled Index Mode	(SP + RWn / RDn × ss + disp 9 / 20)
Register Indirect with Postincrement Mode	(RWn / RDn + +)
Register Indirect with Predecrement Mode	(-- RWn / RDn)
Absolute Addressing Mode	(disp 9 / 13 / 20 / 24)
Immediate Data	imm 4 / 8 / 16 / 32

Most instructions contain one or both of the following two addressing mode fields.

In both addressing mode fields, complicated addressing modes or displacement can be specified using addressing mode extension words.

When the displacement is larger than the addressing mode extension word, prefix is used for extension as necessary.

In register indirect addressing mode, register data is handled as signed and used as a 24-bit address.

Address data used to specify the addressing mode in A format are also handled as signed and used as a 24-bit address.

SEA: Short addressing mode (6 bits)

00rrrr : RBn / RWn / RDn Note: RDn can only specify an even numbered register.

01rrrr : (RWn)

* 10rrrr : (RWn + disp) etc.

11iiii : Quick-Imm(#4). Note: iiii indicates signed.

* : an extension word is required.

When rrrr=0000, constant 0 is used instead of RWO.

LEA: Long addressing mode (8 bits)

~~0000rrrr~~ : RBn / RWn / RDn Note: RDn can only specify an even numbered register.
~~0001rrrr~~ : (RWn)
~~* 0010rrrr~~ : (RWn + disp) etc.
~~# 0011rrrr~~ : Special-Reg
 (Imm., SP, ISP, ESP, PBP, CBP, PSW, IMC, CC)
 rrrr = 0000, 1100, 0001, 0011, 0101, 0111, 1001, 1011, 1110
~~% 0100rrrr~~ : (RWn ++)
~~% 0101rrrr~~ : (-- RWn)
~~& 0110rrr0~~ : (RDn + disp) Note: RDn can only specify an even numbered register.
~~- 0110rrrl~~ : (RDn) Note: RDn can only specify an even numbered register.
~~% 0111rrr0~~ : (RDn ++) Note: RDn can only specify an even numbered register.
~~% 0111rrrl~~ : (--RDn) Note: RDn can only specify an even numbered register.
 1ddddddd : (SP + ddddddd) Note: ddddddd indicates unsigned. (Zero extension)
 * : an extension word is required.
 rrrr=0000, constant 0 is used instead of RWO.
 & : an extension word is required.
 rrr=000, constant 0 is used instead of RDO.
 # : Imm. Data can be added.
 1-or 2-word immediate data is used depending on the operation size.
 % : an address is incremented or decremented by one if in bytes, 2 if in words, and
 4 if in double words, depending on the operation size.

Addressing mode extension word

An addressing mode extension word is added to the basic portion of the addressing modes described above.

The register specified by the bit pattern of the basic portion is represented by Base.

0rrrrssddddddd : (RWn + Base * ss + dddddddd)
 10rrrrssddddddd : (RDn + Base * ss + dddddddd)
 110rrssddddddd : (SpcR + Base * ss + dddddddd)
 111ddddddddd : (Base + dddddddddd)

ss: Scale factor 1(00), 2(01), 4(10), or 8(11) where Brackets () contain bit patterns.

ddd.....ddd : Displacement used after it is sign-extended.

RDn: Can only specify an even numbered register.

SpcR: Constant 0, PC, SP or (undefined and reserved)

rr=00, 11, 10, 01

A scale index is obtained by multiplying the register specified in Base by a scale factor (1/2/4/8) then adding the register contents and the displacement specified by the extension word to the result.

Specifying constant 0 in Base results in PC relative, SP relative, or absolute.

Displacement extension (Prefix function)

Displacements, which can be specified by addressing mode extension words, are either 9 or 13 bits and may not be long enough depending on the operand address.

In such cases, allocating an instruction word which specifies a prefix before the bit pattern of the subject instruction enables a displacement extension (by inputting the excess bits from the upper bits in prefix).

(Note) 1ddddddd in long addressing mode cannot be extended by a prefix.

Instruction word	1	1	0	1	NN	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
NN = 0: prefix extension 0 (pxf0) = 1: prefix extension 1 (pxf1)																

D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0: Displacement data to be extended

How pfx0 and pfx1 are used differs slightly depending on the instruction and operand as follows.

Definition and usage of prefixes

With the TLCS-9000 instruction set, two types of prefixes are used when the displacement size is not large enough. Usage differs as follows.

An instruction can use both pfx0 and pfx1 simultaneously ; either can come first.

Syntax: Operand with (0) can use pfx0 if necessary.

Operand with (1) can use pfx1 if necessary.

ABCD	dst(1),src(0)	Add Decimal with Carry Flag
ADC	dst(1),src(0)	Add with Carry Flag
ADD	dst(1),src(0)	Add
ADD3	dst,src1(1),src2(0)	Add Trinomial
AND	dst(1),src(0)	Logical AND
ANDCF	src(1),num(0)	Bit Logical AND with Carry Flag
BCHG	dst(1),num(0)	1Bit Inversion
BFEX	dst,src(1),num1,num2	Extract Bit Field Unsigned
BFEXS	dst,src(1),num1,num2	Extract Bit Field Signed
BFIN	dst(1),src,num1,num2	Bit Field Insert
BRES	dst(1),num(0)	1Bit Reset
BSOB	dst(1),src(0)	Bit Search for 0 Backward
BSOF	dst(1),src(0)	Bit Search for 0 Forward

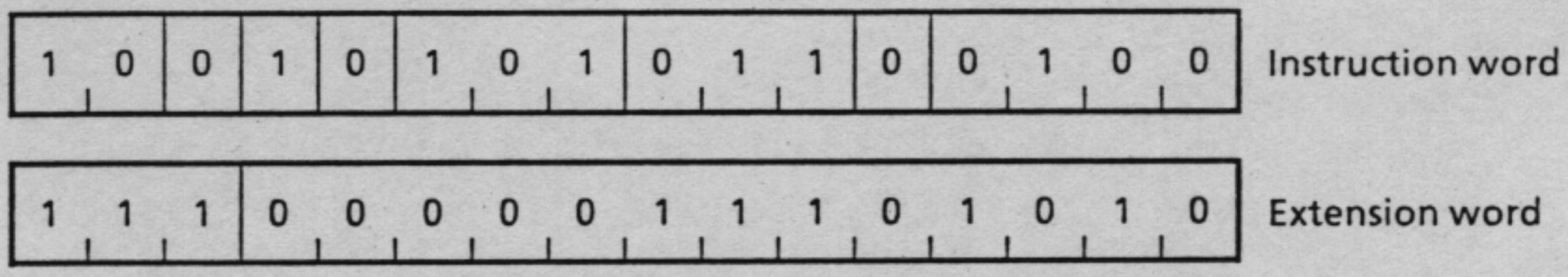
BS1B	dst(1),src(0)	Bit Search for 1 Backward
BS1F	dst(1),src(0)	Bit Search for 1 Forward
BSET	dst(1),num(0)	1Bit Set
BTST	src(1),num(0)	Test 1Bit
CALL	dst(0)	Call Subroutine
CALR	disp(0)	Call Subroutine Relative
CBCD	dst(1),src(0)	Compare Decimal with Carry Flag
CHK	mem(1),src(0)	Check Bound Unsigned
CHKS	mem(1),src(0)	Check Bound Signed
CLR	dst(0)	Clear Operand
CP	dst(1),src(0)	Compare
CPC	dst(1),src(0)	Compare with Carry Flag
CPL	dst(0)	One's-Complement
CPSN	dst(1),src(0),cnts	Compare String (Unmatch Detection)
CPSZ	dst(1),src(0),cnts	Compare String (Match Detection)
DIV	dst(1),src(0)	Unsigned Divide
DIVS	dst(1),src(0)	Signed Divide
DJNZC	dst(0),cond,disp(1)	Decrement, Conditional and Jump Non-zero
DJNZ	dst(0),disp(1)	Decrement and Jump Non-zero
EX	dst1(1),dst2(0)	Exchange
EXTS	dst(0)	Sign Extend
EXTZ	dst(0)	Zero Extend
JP	dst(0)	Jump
JRC	cond,disp(0)	Relative Jump Conditional
JR	disp(0)	Relative Jump
JRBC	num,abs(1),disp(0)	Bit Test Relative Jump and Clear
JRBS	num,abs(1),disp(0)	Bit Test Relative Jump and Set
LD	dst(1),src(0)	Load
LDA	dst(1),src(0)	Load Effective Address
LDCF	src(1),num(0)	Load Bit to Carry Flag
LDS	dst(1),src(0),cnts	Load String
LINK	disp(0)	Link Stack Frame
MAC	dst,src1(1),src2(0)	Unsigned Multiply-and-Add Calculation
MACS	dst,src1(1),src2(0)	Signed Multiply-and-Add Calculation
MAX	dst(1),src(0)	Unsigned Maximum Value
MAXS	dst(1),src(0)	Signed Maximum Value
MIN	dst(1),src(0)	Unsigned Minimum Value
MINS	dst(1),src(0)	Signed Minimum Value
MIRR	dst(0)	Mirror Exchange
MUL	dst(1),src(0)	Unsigned Multiply
MULS	dst(1),src(0)	Signed Multiply
NEG	dst(0)	Negate
OR	dst(1),src(0)	Logical OR
ORCF	src(1),num(0)	Bit Logical OR with Carry Flag
POP	dst(0)	Pop from Stack
PUSH	src(0)	Push to Stack
PUSHA	dst(0)	Push Effective Address to Stack
RETD	disp(0)	Return and Delete Parameter Area
RL	dst(1),num(0)	Rotate Left with Carry Flag
RLC	dst(1),num(0)	Rotate Left without Carry Flag
RLM	dst1(1),dst2,num(0)	Rotate Left Multi Bit

RR	dst(1),num(0)	Rotate Right with Carry Flag
RRC	dst(1),num(0)	Rotate Right without Carry Flag
RRM	dst1(1),dst2,num(0)	Rotate Right Multi Bit
RVBY	dst(0)	Reverse Byte
SBC	dst(1),src(0)	Subtract with Carry Flag
SBCD	dst(1),src(0)	Subtract Decimal with Carry Flag
SLA	dst(1),num(0)	Arithmetic Shift Left
SLL	dst(1),num(0)	Logical Shift Left
SRA	dst(1),num(0)	Arithmetic Shift Right
SRL	dst(1),num(0)	Logical Shift Right
STCF	dst(1),num(0)	Bit Transfer from Carry Flag (??)
SUB	dst(1),src(0)	Subtract
SUB3	dst,src1(1),src2(0)	Subtract Trinomial
TJP	dst(0)	Table Jump
TSET	dst(1),num(0)	Test and Set 1 Bit
TST	dst(0)	Operand Test
XOR	dst(1),src(0)	Exclusive OR
XORCF	src(1),num(0)	Bit Exclusive OR with Carry Flag

Example of addressing mode field organization

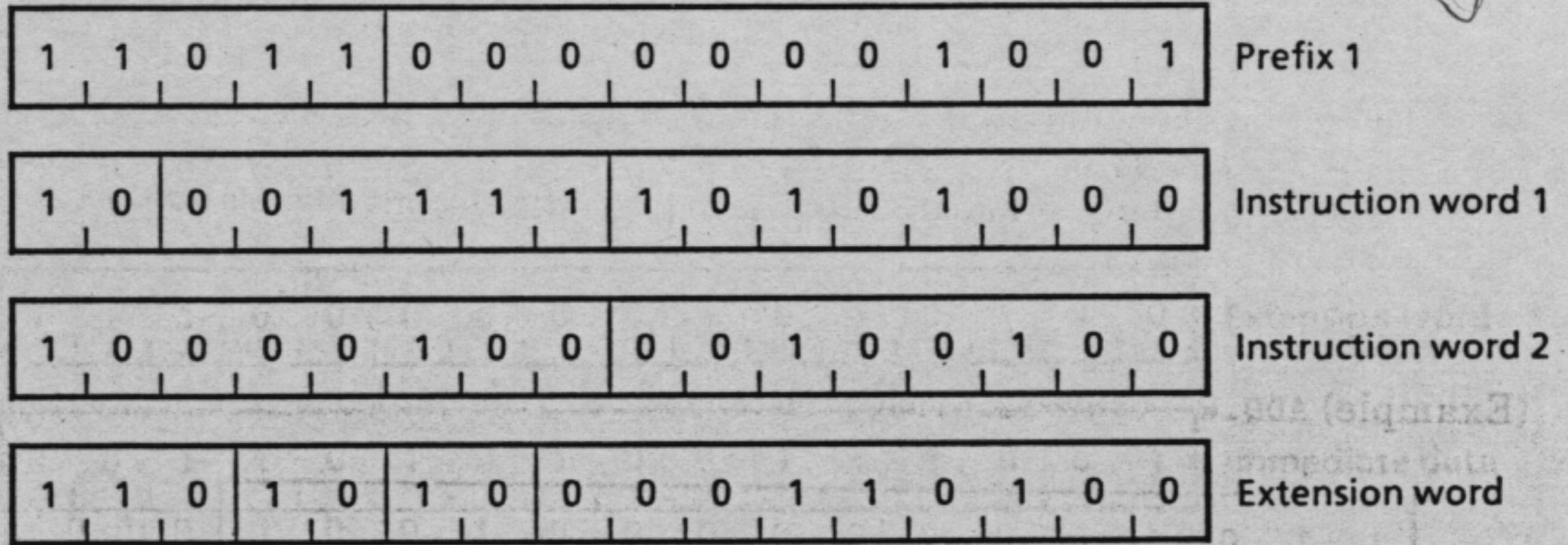
Below is an example of a typical addressing mode field organization.

(Example) ADD.W (RW4+0EAh),RW7 ... S format



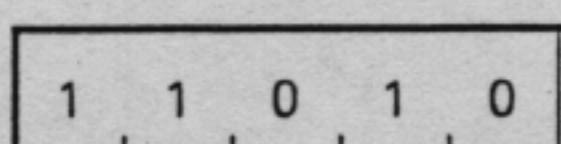
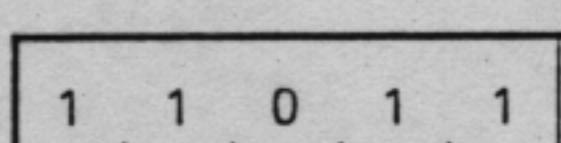
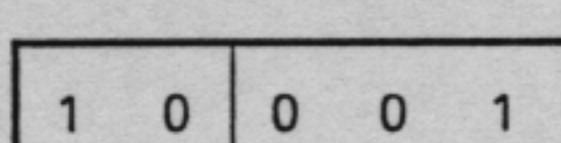
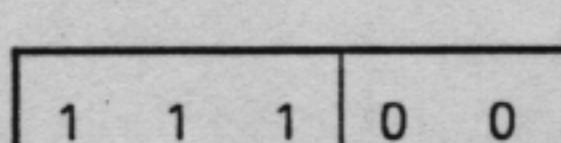
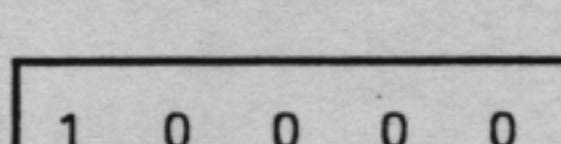
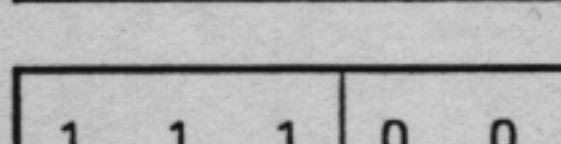
Point: Extension word can also be added to short addressing mode.

(Example) ADD.W (SP+RW4*4+1234h),(SP+28h) ... G format



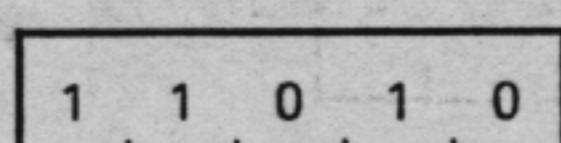
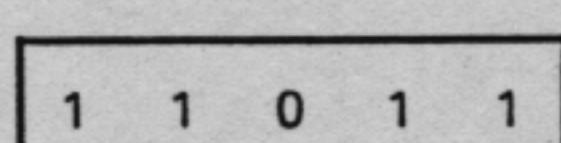
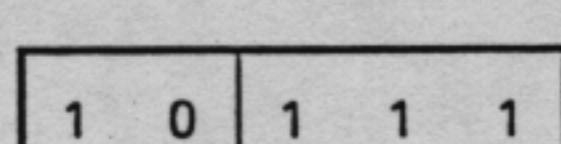
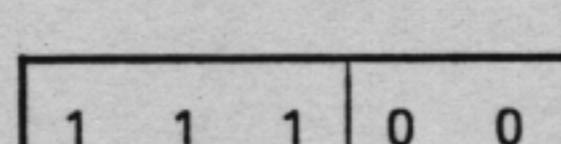
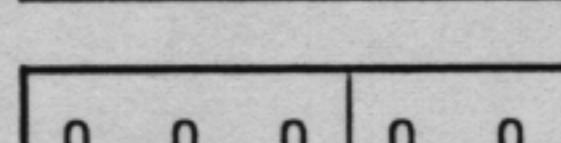
Point: Prefix must be added before instruction word 1.

(Example) ADD.W:G (RW4+12345h),(10028h) ... G format

 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0	Prefix 0
 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1	Prefix 1
 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0	Instruction word 1
 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0	Extension word
 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0	Instruction word 2
 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1	Extension word

Point: Extension word must be added immediately after the basic portion of addressing mode.
When constant 0 is specified in the addressing mode basic portion, absolute address is set.

(Example) ADD.W (RW4+12345h),(10028h) ... A format

 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0	Prefix 0
 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1	Prefix 1
 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 0	Instruction word 1
 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1	Extension word
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0	Instruction word 2

Point: Total instruction word is 1 word shorter than the above example in G format.

(Example) ADD.W:G (RW4+56h),3579h ... G format

1 0 0 0 1 1 1 0 0 1 1 0 0 0 0	Instruction word 1
0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1	Immediate data
1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0	Instruction word 2
1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0	Extension word

Point: Extension word must be added immediately after the addressing mode basic portion.

(Example) ADD.W (RW4+56h),3579h ... I format

The diagram illustrates a 16-bit instruction word divided into three fields:

- Instruction word:** The first 8 bits (00000000).
- Extension word:** The next 6 bits (001001).
- Immediate data:** The last 2 bits (10).

Point: Total instruction word is 1 word shorter than the above example in G format.

(Example) JRC NZ,\$-1234h Prefix is added because disp is larger than 9 bits.

1	1	0	1	0	1	1	1	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Prefix 0

Point: Displacement is sign-extended after linked to prefix. \$ symbol indicates PC relative .

All instructions are listed below in alphabetical order.
Detailed description for each of them follows.

1	ABCD	dst,src	: Add Decimal with Carry Flag
2	ADC	dst,src	: Add with Carry Flag
3	ADD	dst,src	: Add
4	ADD3	dst,src1,src2	: Add Trinomial
5	AND	dst,src	: Logical AND
6	ANDCF	src,num	: Bit Logical AND with Carry Flag
7	BCHG	dst,num	: Bit Change
8	BFEX	dst,src,num1,num2	: Extract Bit Field Unsigned
9	BFEXS	dst,src,num1,num2	: Extract Bit Field Signed
10	BFIN	dst,src,num1,num2	: Bit Field Insert
11	BRES	dst,num	: Bit Reset
12	BSOB	dst,src	: Bit Search for 0 Backward
13	BSOF	dst,src	: Bit Search for 0 Forward
14	BS1B	dst,src	: Bit Search for 1 Backward
15	BS1F	dst,src	: Bit Search for 1 Forward
16	BSET	dst,num	: Bit Set
17	BTST	src,num	: Test Bit
18	CALL	dst	: Call Subroutine
19	CALR	disp	: Call Subroutine Relative
20	CBCD	dst,src	: Compare Decimal with Carry Flag
21	CCF		: Reverse Carry Flag
22	CHK	mem,src	: Check Bound Unsigned
23	CHKS	mem,src	: Check Bound Signed
24	CLR	dst	: Clear Operand
25	CP	dst,src	: Compare
26	CPC	dst,src	: Compare with Carry Flag
27	CPL	dst	: Ones-Complement
28	CPSN	dst,src,conts	: Compare String (Unmatch detection)
29	CPSZ	dst,src,conts	: Compare String (Match detection)
30	CSF		: Invert Sign Flag
31	CVF		: Invert Overflow Flag
32	CZF		: Invert Zero Flag
33	DI		: Disable Interrupt
34	DIV	dst,src	: Unsigned Divide
35	DIVS	dst,src	: Signed Divide
36	DJNZ	dst,disp	: Decrement and Jump Non-zero
37	DJNZC	dst,cond,disp	: Decrement, Conditional and Jump Non-zero
38	EI		: Enable Interrupt
39	EX	dst1,dst2	: Exchange
40	EXTS	dst	: Sign Extend
41	EXTZ	dst	: Zero Extend
42	HALT		: Halt CPU
43	JP	dst	: Jump

44	JR	disp	: Relative Jump
45	JRBC	num,abs,disp	: Bit Test Relative Jump and Clear
46	JRBS	num,abs,disp	: Bit Test Relative Jump and Set
47	JRC	cond,disp	: Relative Jump Conditional
48	LD	dst,src	: Load
49	LDA	dst,src	: Load Effective Address
50	LDCF	src,num	: Load Bit to Carry Flag
51	LDS	dst,src,cnts	: Load String
52	LINK	disp	: Link Stack Frame
53	MAC	dst,src1,src2	: Unsigned Multiply-and-Add Operation
54	MACS	dst,src1,src2	: Signed Multiply-and-Add Operation
55	MAX	dst,src	: Unsigned Maximum Value
56	MAXS	dst,src	: Signed Maximum Value
57	MIN	dst,src	: Unsigned Minimum Value
58	MINS	dst,src	: Signed Minimum Value
59	MIRR	dst	: Mirror Exchange
60	MUL	dst,src	: Unsigned Multiply
61	MULS	dst,src	: Signed Multiply
62	NEG	dst	: Negate
63	NOP		: No Operation
64	OR	dst,src	: Logical OR
65	ORCF	src,num	: Bit Logical OR with Carry Flag
66	POP	dst	: Pop from Stack
67	PUSH	src	: Push to Stack
68	PUSHA	dst	: Push Effective Address to Stack
69	RCF		: Reset Carry Flag
70	RET		: Return
71	RETD	disp	: Return and Delete Parameter Area
72	RETI		: Return from Interrupt
73	RETS		: Return from Single Step Interrupt
74	RL	dst,num	: Rotate Left with Carry Flag
75	RLC	dst,num	: Rotate Left without Carry Flag
76	RLM	dst1,dst2,num	: Rotate Left Multi Bit
77	RR	dst,num	: Rotate Right with Carry Flag
78	RRC	dst,num	: Rotate Right without Carry Flag
79	RRM	dst1,dst2,num	: Rotate Right Multi Bit
80	RSF		: Reset Sign Flag
81	RVBY	dst	: Reverse Byte
82	RVF		: Reset Overflow Flag Reset
83	RZF		: Reset Zero Flag
84	SBC	dst,src	: Subtract with Carry Flag
85	SBCD	dst,src	: Subtract Decimal with Carry Flag
86	SCF		: Set Carry Flag
87	SLA	dst,num	: Arithmetic Shift Left
88	SLL	dst,num	: Logical Shift Left
89	SRA	dst,num	: Arithmetic Shift Right
90	SRL	dst,num	: Logical Shift Right
91	SSF		: Set Sign Flag
92	STCF	dst,num	: Bit Transfer from Carry Flag

93	SUB	dst,src	:	Subtract
94	SUB3	dst,src1,src2	:	Subtract Trinomial
95	SVF		:	Set Overflow Flag
96	SWI	vec	:	Software Interrupt
97	SZF		:	Set Zero Flag
98	TJP	dst	:	Table Jump
99	TSET	dst,num	:	Test and Set Bit
100	TST	dst	:	Operand Test
101	UNLK		:	Unlink Stack Frame
102	XOR	dst,src	:	Exclusive OR
103	XORCF	src,num	:	Bit Exclusive OR with Carry Flag

1. List of Instruction Formats

Instruction	Addressing	Number of minimum instruction words
ABCD	G,A	2
ADC	G,A	2
ADD	S,G,I,A	2
ADD3		2
AND	S,G,I,A	1
ANDCF	G,A	2
BCHG	S,G,A	1
BFEX		2
BFEXS		2
BFIN		2
BRES	S,G,A	1
BSOB	G,A	2
BSOF	G,A	2
BS1B	G,A	2
BS1F	G,A	2
BSET	S,G,A	1
BTST	S,G,A	1
CALL		1
CALR		1
CBCD	G,A	2
CCF		1
CHK	G,A	2
CHKS	G,A	2
CLR		1
CP	S,G,I,A	1
CPC	G,A	2
CPL		1
CPSN		2
CPSZ		2
CSF		1
CVF		1
CZF		1
DI		1
DIV	G,A	2
DIVS	G,A	2

DJNZC		2
DJNZ		2
EI		1
EX	S,G,A	1
EXTS		1
EXTZ		1
HALT		1
JP		1
JRC		1
JR		1
JRBC		2
JRBS		2
LD	S,G,I,A	1
LDA		2
LDCF	G,A	2
LDS		2
LINK		1
MAC		2
MACS		2
MAX	G,A	2
MAXS	G,A	2
MIN	G,A	2
MINS	G,A	2
MIRR		1
MUL	G,A	2
MULS	G,A	2
NEG		1
NOP		1
OR	S.G,I,A	1
ORCF	G,A	2
POP		1
PUSH		1
PUSHA		1
RCF		1
RET		1
RETD		1
RETI		1
RETS		1
RL	S,G,A	1
RLC	S,G,A	1
RLM		2
RR	S,G,A	1
RRC	S,G,A	1
RRM		2
RSF		1
RVBY		1
RVF		1
RZF		1
SBC	G,A	2

SBCD	G,A	2
SCF		1
SLA	S,G,A	1
SLL	S,G,A	1
SRA	S,G,A	1
SRL	S,G,A	1
SSF		1
STCF	G,A	2
SUB	S,G,I,A	1
SUB3		2
SVF		1
SWI		1
SZF		1
TJP		1
TSET	G,A	2
TST		1
UNLK		1
XOR	S,G,I,A	1
XORCF	G,A	2

2. I Format List

ADD	S,G,I,A	1
AND	S,G,I,A	1
CP	S,G,I,A	1
LD	S,G,I,A	1
OR	S,G,I,A	1
SUB	S,G,I,A	1
XOR	S,G,I,A	1

3. S Format List

ADD	S,G,I,A	1
AND	S,G,I,A	1
BCHG	S,G,A	1
BRES	S,G,A	1
BSET	S,G,A	1
BTST	S,G,A	1
CP	S,G,I,A	1
EX	S,G,A	1
LD	S,G,I,A	1
OR	S,G,I,A	1
RL	S,G,A	1
RLC	S,G,A	1
RR	S,G,A	1
RRC	S,G,A	1
SLA	S,G,A	1
SLL	S,G,A	1
SRA	S,G,A	1

SRL	S,G,A	1
SUB	S,G,I,A	1
XOR	S,G,I,A	1

4. G Format List

ABCD	G,A	2
ADC	G,A	2
ADD	S,G,I,A	1
AND	S,G,I,A	1
ANDCF	G,A	2
BCHG	S,G,A	1
BRES	S,G,A	1
BSOB	G,A	2
BSOF	G,A	2
BS1B	G,A	2
BS1F	G,A	2
BSET	S,G,A	1
BTST	S,G,A	1
CBCD	G,A	2
CHK	G,A	2
CHKS	G,A	2
CP	S,G,I,A	1
CPC	G,A	2
DIV	G,A	2
DIVS	G,A	2
EX	S,G,A	1
LD	S,G,I,A	1
LDCF	G,A	2
MAX	G,A	2
MAXS	G,A	2
MIN	G,A	2
MINS	G,A	2
MUL	G,A	2
MULS	G,A	2
OR	S,G,I,A	1
ORCF	G,A	2
RL	S,G,A	1
RLC	S,G,A	1
RR	S,G,A	1
RRC	S,G,A	1
SBC	G,A	2
SBCD	G,A	2
SLA	S,G,A	1
SLL	S,G,A	1
SRA	S,G,A	1
SRL	S,G,A	1

STCF	G,A	2
SUB	S,G,I,A	1
TSET	G,A	2
XOR	S,G,I,A	1
XORCF	G,A	2

5. A Format List

ABCD	G,A	2
ADC	G,A	2
ADD	S,G,I,A	1
AND	S,G,I,A	1
ANDCF	G,A	2
BCHG	S,G,A	1
BRES	S,G,A	1
BSOB	G,A	2
BSOF	G,A	2
BS1B	G,A	2
BS1F	G,A	2
BSET	S,G,A	1
BTST	S,G,A	1
CBCD	G,A	2
CHK	G,A	2
CHKS	G,A	2
CP	S,G,I,A	2
CPC	G,A	2
DIV	G,A	2
DIVS	G,A	2
EX	S,G,A	1
LD	S,G,I,A	1
LDCF	G,A	2
MAX	G,A	2
MAXS	G,A	2
MIN	G,A	2
MINS	G,A	2
MUL	G,A	2
MULS	G,A	2
OR	S,G,I,A	1
ORCF	G,A	2
RL	S,G,A	1
RLC	S,G,A	1
RR	S,G,A	1
RRC	S,G,A	1
SBC	G,A	2
SBCD	G,A	2
SLA	S,G,A	1
SLL	S,G,A	1
SRA	S,G,A	1
SRL	S,G,A	1
STCF	G,A	2
SUB	S,G,I,A	1

TSET	G,A	2
XOR	S,G,I,A	1
XORCF	G,A	2

6. Other Format List

ADD3		2
BFEX		2
BFEXS		2
BFIN		2
CALL		1
CALR		1
CCF		1
CLR		1
CPL		1
CPSN		2
CPSZ		2
CSF		1
CVF		1
CZF		1
DJNZC		2
DJNZ		2
EI		1
EXTS		1
EXTZ		1
JP		1
JRC		1
JR		1
JRBC		1
JRBS		2
HALT		1
LDA		2
LDS		2
LINK		2
MAC		1
MACS		2
MIRR		2
NEG		1
NOP		1
POP		1
PUSH		1
PUSHA		1
RCF		1
RET		1

RETD	1
RETI	1
RETS	1
RLM	2
RRM	2
RSF	1
RVBY	1
RVF	1
RZF	1
SCF	1
SSF	1
SUB3	2
SVF	1
SWI	1
SZF	1
TJP	1
TST	1
UNLK	1

7. Instructions Classified according to Operation

Load		
CLR		1
LD	S,G,I,A	1
LDA		2
POP		1
PUSH		1
PUSHA		1
Exchange		
EX	S,G,A	1
MIRR		1
RVBY		1
Block load/block search		
CPSN		2
CPSZ		2
LDS		2
Arithmetic operation		
ABCD	G,A	2
ADC	G,A	2
ADD	S,G,I,A	2
ADD3		2
CBCD	G,A	2
CHK	G,A	2
CHKS	G,A	2
CP	S,G,I,A	1
CPC	G,A	2
DIV	G,A	2
DIVS	G,A	2
EXTS		1

EXTZ		1
MAC		2
MACS		2
MAX	G,A	2
MAXS	G,A	2
MIN	G,A	2
MINs	G,A	2
MUL	G,A	2
MULS	G,A	2
NEG		1
SBC	G,A	2
SBCD	G,A	2
SUB	S,G,I,A	1
SUB3		2
TST		1

Logical operation

AND	S,G,I,A	1
CPL		1
OR	S,G,I,A	1
XOR	S,G,I,A	1

Bit operation

ANDCF	G,A	2
BCHG	S,G,A	1
BFEX		2
BFE XS		2
BFIN		2
BRES	S,G,A	1
BSOB	G,A	2
BSOF	G,A	2
BS1B	G,A	2
BS1F	G,A	2
BSET	S,G,A	1
BTST	S,G,A	1
CCF		1
CSF		1
CVF		1
CZF		1
LDCF	G,A	2
ORCF	G,A	2
RCF		1
RSF		1
RVF		1
RZF		1
SCF		1
SSF		1
STCF	G,A	2
SVF		1
SZF		1

TSET	G,A	2
XORCF	G,A	2

Special operation and CPU control

DI		1
EI		1
HALT		1
LINK		1
NOP		1
SWI		1
UNLK		1

Rotate and shift

RL	S,G,A	1
RLC	S,G,A	1
RLM		2
RR	S,G,A	1
RRC	S,G,A	1
RRM		2
SLA	S,G,A	1
SLL	S,G,A	1
SRA	S,G,A	1
SRL	S,G,A	1

Jump, call, and return

CALL		1
CALR		1
DJNZC		2
DJNZ		2
JP		1
JRC		1
JR		1
JRBC		2
JRBS		2
RET		1
RETD		1
RETI		1
RETS		1
TJP		1

ABCD dst, src : Add Decimal with Carry Flag

Operation : $dst \leftarrow dst + src + C$

Description : Decimal-adds the contents of dst, src, and carry flag C, then loads the result in dst.

Instruction format :

G format :

ABCD.@:G LEA, #8
ABCD.@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	0	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

ABCD.@:A MEM, SEA
ABCD.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c Operation size : B W D

*	-	-	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The Z flag is set to 1 when the Z flag = 1 and the operation result = 0; otherwise, cleared to zero.

MSB of sign-extended immediate data must be "0". If immediate data need to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0 ; Byte operation - 00~99H

Word or Double Word operation - 00~79H

A format, DD=0 and SEA=Quick Imm : 0~7

ADC dst, src : Add with Carry Flag

Operation : $dst \leftarrow dst + src + C$

Description : Adds the contents of dst and src and carry flag C, then loads the result in dst.

Instruction format :

G format :

ADC.:@:G LEA,#8
ADC.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	1	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

ADC.:@:A MEM,SEA
ADC.:@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
1	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change:

*	*	*	*
---	---	---	---

Operation size : B W D

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The Z flag is set to 1 when the Z flag = 1 and the operation result = 0; otherwise, cleared to zero.

ADD dst,src : Add

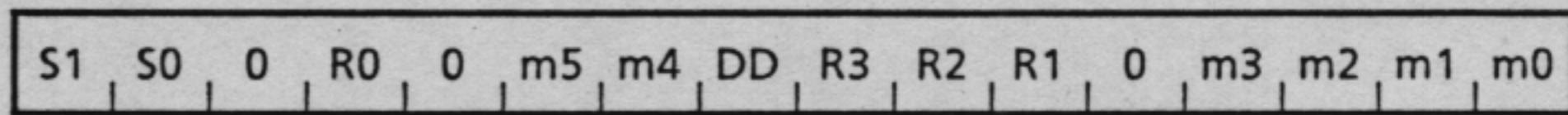
Operation : $dst \leftarrow dst + src$

Description : Adds the contents of dst and src, then loads the result in dst.

Instruction format :

S format :

ADD.@@:S Reg,SEA
ADD.@@:S SEA,Reg

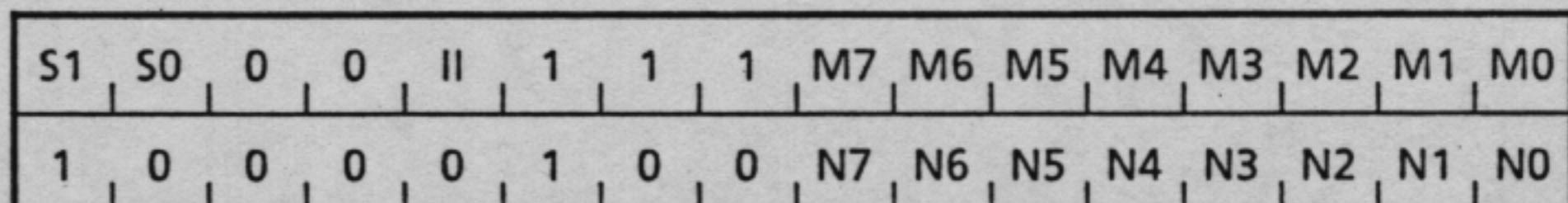


When DD = 0, dst is specified by R<3:0> and src by m<5:0>

When DD = 1, dst and src are exchanged.

G format :

ADD.@@:G LEA,#8
ADD.@@:G LEA,LEA



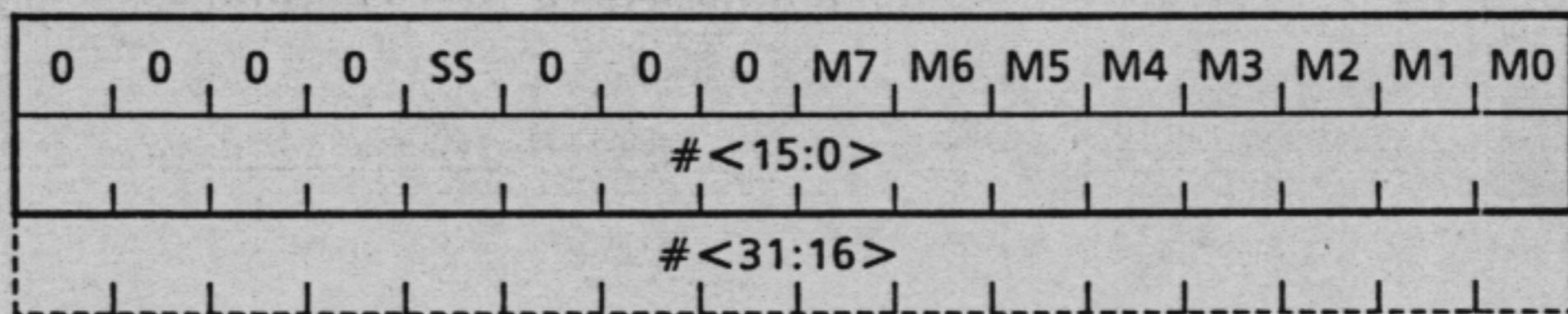
M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

I format :

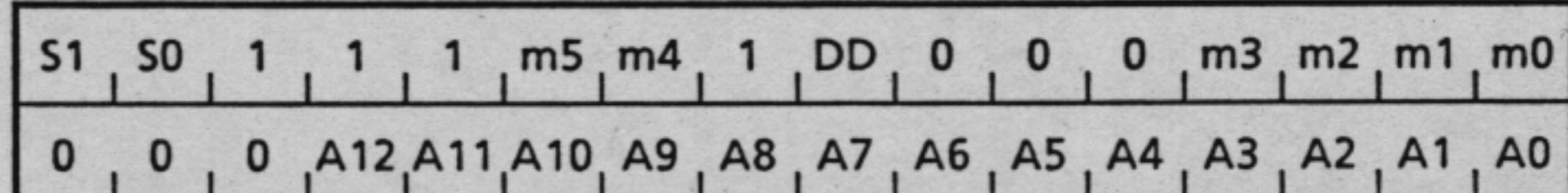
ADD.@@:I LEA,#16
ADD.@@:I LEA,#32



M<7:0> = dst

A format :

ADD.@@:A MEM,SEA
ADD.@@:A SEA,MEM



When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

*	*	*	*
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.
m5m4 = 11 (quick immediate) and m5m4 = 00 (register direct) in S format
allowed only when DD = 0.
Immediate mode inhibited as dst addressing mode.
m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

ADD3 dst,src1,src2 : Add Trinomial

Operation : $dst \leftarrow src1 + src2$

Description : Adds the contents of src1 and src2, then loads the result in dst.

Instruction format :

ADD3.0 Reg,LEA,#8
ADD3.0 Reg,LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
0	R3	R2	R1	R0	0	0	0	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = src2 When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = src1, long addressing mode only

R<3:0> = dst

Flag status change: Z S V C Operation size : B W D

*	*	*	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as src1 addressing mode.

AND dst, src : Logical AND

✓

Operation : $dst \leftarrow dst \text{ AND } src$

Description : ANDs the contents of dst and src, then loads the result in dst.

Instruction format :

S format :

AND.:@:S Reg1,Reg2

S1	S0	1	1	1	0	0	0	T3	T2	T1	T0	R3	R2	R1	RO
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

T<3:0> = dst

R<3:0> = src

G format :

AND.:@:G LEA,#8
AND.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	0	1	0	0	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

I format:

AND.:@:I LEA,#16
AND.:@:I LEA,#32

0	0	0	0	SS	1	0	0	M7	M6	M5	M4	M3	M2	M1	M0
								#<15:0>							
									#<31:16>						

M<7:0> = dst

A format :

AND.:@:A MEM,SEA
AND.:@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c Operation size : b w d

*	*	0	0
---	---	---	---

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.

Immediate mode inhibited as dst addressing mode.

$m_5m_4 = 11$ (quick immediate) in A format allowed only when DD = 0.

ANDCF src, num : Bit Logical AND with Carry Flag

Operation : $C \leftarrow C \text{ AND } \text{src} < \text{num} >$

Description : ANDs the contents of carry flag C with bit num of src, then loads the result in carry flag C.

Instruction format :

G format :

ANDCF. @:G LEA, #8
ANDCF. @:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	1	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = src, long addressing mode only

A format :

ANDCF. @:A MEM, SEA
ANDCF. @:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
1	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, src is memory specified by A<12:0> and num by m<5:0>.
When DD = 1, num and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4 bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

BCHG dst, num : Bit Change

V

Operation : $dst < num > \leftarrow \text{inverted value of } dst < num >$

Description : Inverts the bit num value in dst.

Instruction format :

S format :

BCHG.@:S Reg, num

S1	S0	0	1	1	1	1	1	0	B2	B1	B0	R3	R2	R1	R0
----	----	---	---	---	---	---	---	---	----	----	----	----	----	----	----

B<2:0> = num (When the operation size is word, num-8.)

R<3:0> = dst

In S format, the operation size is either byte or word.

G format :

BCHG.@:G LEA, #8
BCHG.@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	0	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

BCHG.@:A MEM, SEA
BCHG.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
0	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and num by m<5:0>.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

BFEX dst, src, num1, num2 : Extract Bit Field Unsigned

U

Operation : $dst \leftarrow src < num1 + num2 - 1 : num1 >$

Description : Extracts the bit field specified by bit offset num1 at address src and bit size num2, then loads the result without sign in register dst.

Instruction format :

BFEX. @ *Reg, LEA
num1, num2

0	1	1	1	0	0	S1	S0	M7	M6	M5	M4	M3	M2	M1	M0
0	R3	R2	R1	R0	0	0	P3	P2	P1	P0	0	W3	W2	W1	W0

M<7:0> = src, long addressing mode only

R<3:0> = dst, register only

P<3:0> = num1 (bit offset)

W<3:0> = num2 (bit size - 1)

=#-1

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : The operation size indicates the src operand size. dst always consists of words.

The bit field is handled as unsigned. Thus, when bit size num2 is smaller than register dst, data is zero-extended then loaded in register dst.

Immediate mode inhibited as src addressing mode.

When the operation size is double word and the bit field corresponding to the upper word is to be operated on:

- For registers, specify the value larger than the register number by 1 and execute in words.
- For memory, specify the value larger than the normally specified address by 2 and execute in words.

To enable word execution, bit offset is restricted to 0 to 15.

Write 1 to 16 in num2 in Assembler, then 0 to 15 obtained by decrementing by 1 is stored in the instruction codes.

When the offset (num1) plus bit size (num2) exceeds the src operand size, the exceeded part is regarded as 0s.

BFEXS dst, src, num1, num2 : Extract Bit Field Signed

Operation : $dst \leftarrow src < num1 + num2 - 1 : num1 >$

Description : Extracts the bit field specified by bit offset num1 at address src and bit size num2, then loads the result with sign in register dst.

Instruction format :

BFEXS. @ Reg, LEA num1, num2	0 1 1 1 0 0 S1 S0 M7 M6 M5 M4 M3 M2 M1 M0
	0 R3 R2 R1 R0 1 0 P3 P2 P1 P0 0 W3 W2 W1 W0

M<7:0> = src, long addressing mode only

R<3:0> = dst, register only

P<3:0> = num1 (bit offset)

W<3:0> = num2 (bit size -1)

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : The operation size indicates the src operand size. dst always consists of words.

The bit field is handled as signed. Thus, when bit size num2 is smaller than register dst, data is sign-extended, then loaded in register dst.

Immediate mode inhibited as src addressing mode.

When the operation size is double word and the bit field corresponding to the upper word is to be operated on:

- For registers, specify the value larger than the register number by 1 and execute in words.

- For memory, specify the value larger than the normally specified address by 2 and execute in words.

To enable word execution, bit offset is restricted to 0 to 15.

Write 1 to 16 in num2 in Assembler, then 0 to 15 obtained by subtracting 1 from the num2 is stored in instruction codes.

When the offset (num1) plus bit size (num2) exceeds the src operand size, the exceeded part is regarded as 0s.

BFIN dst, src, num1, num2 : Bit Field Insert

Operation : $dst <num1+num2-1: num1> \leftarrow src <num2-1: 0>$

Description : Loads the bit string specified by bit size num2 of register src in the bit field specified by bit offset num1 and bit size num2 at address dst.

Instruction format :

BFIN. @ LEA, Reg
num1, num2

0	1	1	1	0	0	S1	S0	M7	M6	M5	M4	M3	M2	M1	M0
1	R3	R2	R1	R0	0	0	P3	P2	P1	P0	0	W3	W2	W1	W0

$M <7:0>$ = dst, long addressing mode only

$R <3:0>$ = src, register only

$P <3:0>$ = num1 (bit offset)

$W <3:0>$ = num2 (bit size -1)

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : The operation size indicates the dst operand size. src always consists of words.

Immediate mode inhibited as src addressing mode.

When the operation size is double word and the bit field corresponding to the upper word is to be operated on:

- For registers, specify the value larger than the register number by 1 and execute in words.
- For memory, specify the value larger than the normally specified address by 2 and execute in words.

To enable word execution, bit offset is restricted to 0 to 15.

Write 1 to 16 in num2 in Assembler, then 0 to 15 obtained by subtracting 1 from the written value is stored in instruction codes.

When the offset (num1) plus bit size (num2) exceeds the dst operand size, the exceeded part is discarded and nothing remains in dst.

BRES dst, num : Bit Reset

Operation : $dst < num > \leftarrow 0$

Description : Resets bit num in dst.

Instruction format :

S format :

BRES.@@S Reg, num

S1	S0	0	1	0	1	1	1	0	B2	B1	B0	R3	R2	R1	RO
----	----	---	---	---	---	---	---	---	----	----	----	----	----	----	----

B<2:0> = num (When the operation size is word, num-8.)

R<3:0> = dst

In S format, the operation size is either byte or word.

G format :

BRES.@@G LEA, #8
BRES.@@G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	0	1	0	0	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

BRES.@@A MEM, SEA
BRES.@@A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and num by m<5:0>.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

BS0B dst, src : Bit Search for 0 Backward

Operation : $dst \leftarrow$ operation result when searching src for the first 0 backward.

Description : Searches the bit pattern of src for the first 0 backward (from MSB to LSB), then loads the number of bit where the first 0 is found in dst.

Instruction format :

G format :

BS0B.@@:G LEA, #8
BS0B.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	1	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

BH0B.@@:A MEM, SEA
BS0B.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
1	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	*	-
---	---	---	---

○	○	○
---	---	---

Notes : If 0 was not found in the searched bit pattern, the contents of dst are set to indeterminate and the V flag is set to 1; otherwise, reset to 0.

Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

BSOF dst, src : Bit Search for 0 Forward

Operation : $dst \leftarrow$ operation result when searching src for the first 0 forward.

Description : Searches the bit pattern of src for the first 0 forward (from LSB to MSB) and loads the number of bit where the first 0 is found in dst.

Instruction format :

G format :

BSOF.@@:G LEA, #8
BSOF.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	1	1	0	1	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

BSOF.@@:A MEM, SEA
BSOF.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
1	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$ and src by $m<5:0>$.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	*	-
---	---	---	---

O	O	O
---	---	---

Notes : If 0 was not found in the searched bit pattern, the contents of dst are set to indeterminate and the V flag is set to 1; otherwise, reset to 0.

Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

BS1B dst, src : Bit Search for 1 Backward

Operation : $dst \leftarrow$ operation result when searching src for the first 1 backward.

Description : Searches the bit pattern of src for the first 1 backward (from MSB to LSB), then loads the number of bit where the first 1 is found in dst.

Instruction format :

G format :

BS1B.@@:G LEA, #8
BS1B.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	1	1	1	0	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

BS1B.@@:A MEM, SEA
BS1B.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
1	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$ and src by $m<5:0>$.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	*	-
---	---	---	---

○	○	○
---	---	---

Notes : If 1 was not found in the searched bit pattern, the contents of dst are set to indeterminate and the V flag is set to 1; otherwise, reset to 0.

Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

BS1F dst, src : Bit Search for 1 Forward

Operation : $dst \leftarrow$ operation result when searching src for the first 1 forward.

Description : Searches the bit pattern of src for the first 1 forward (from LSB to MSB), then loads the number of bit where the first 1 is found in dst.

Instruction format :

G format :

BS1F.@@:G LEA, #8
BS1F.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	1	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

BS1F.@@:A MEM, SEA
BS1F.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
1	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$ and src by $m<5:0>$.

When DD = 1, dst and src are exchanged.

Flag status change: Z S V C Operation size : B W D

-	-	*	-
---	---	---	---

○	○	○
---	---	---

Notes : If 1 was not found in the searched bit pattern, the contents of dst are set to indeterminate and the V flag is set to 1; otherwise, reset to 0.

Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

BSET dst, num : Bit Set

Operation : $dst < num > \leftarrow 1$

Description : Sets bit num in dst to 1.

Instruction format :

S format :

BSET.@:S Reg, num

S1	S0	0	1	0	1	1	1	1	B2	B1	B0	R3	R2	R1	R0
----	----	---	---	---	---	---	---	---	----	----	----	----	----	----	----

B<2:0> = num (When the operation size is word, num-8.)

R<3:0> = dst

In S format, the operation size is either byte or word.

G format :

BSET.@:G LEA, #8
BSET.@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	0	1	0	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

BSET.@:A MEM, SEA
BSET.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
0	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and num by m<5:0>.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

BTST src, num : Bit Test

✓

Operation : $Z \leftarrow \text{inverted value of } \text{src} < \text{num} >$

Description : Loads the inverted value of bit num of src in the Z flag.

Instruction format :

S format :

BTST.@:S Reg, num

S1	S0	0	1	1	1	1	1	1	B2	B1	B0	R3	R2	R1	R0
----	----	---	---	---	---	---	---	---	----	----	----	----	----	----	----

B<2:0> = num (When the operation size is word, num-8.)

R<3:0> = src

In S format, the operation size is either byte or word.

G format :

BTST.@:G LEA, #8
BTST.@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	1	0	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = src, long addressing mode only

A format :

BTST.@:A MEM, SEA
BTST.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	1	m3	m2	m1	m0
0	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and num by m<5:0>.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c

Operation size : B W D

*	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

CALL dst : Call Subroutine

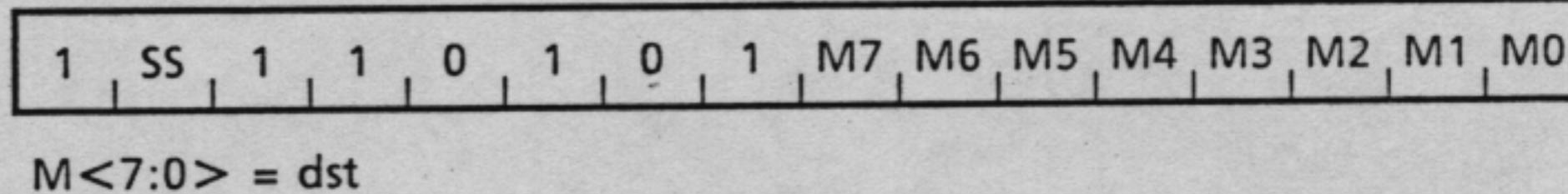
✓

Operation : $SP \leftarrow SP - 4, (SP) \leftarrow PC, PC \leftarrow$ effective addresses of dst

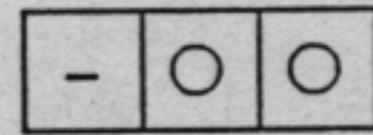
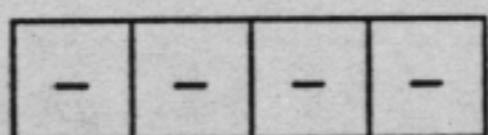
Description : Saves the contents of the program counter in the stack area and jumps to the dst effective address.

Instruction format :

CALL. @ LEA



Flag status change: z s v c Operation size : B W D



Notes : $SP \leftarrow SP - 4$ is executed after the dst effective address is calculated.
 Register direct or immediate mode inhibited as dst addressing mode.
 The operation size is employed to increment or decrement an address by $(RWn++)$, $(-RWn)$, $(RDn++)$ or $(-RDn)$ in addressing mode.
 When the effective address of dst is odd numbered, jumps to an even numbered address obtained by rounding off the least significant bit to 0.

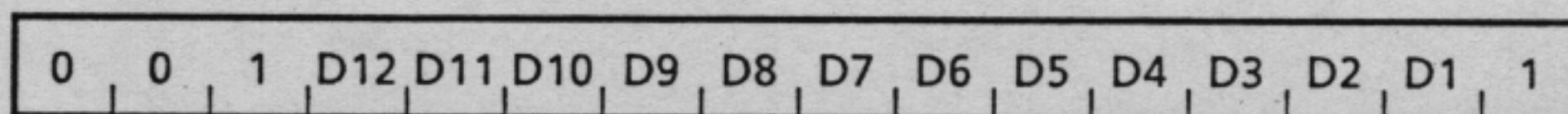
CALR disp : Call Subroutine Relative

Operation : $SP \leftarrow SP - 4, (SP) \leftarrow PC, PC \leftarrow PC + disp$

Description : Saves the contents of the program counter in the stack area and restarts the program from $PC + disp$.

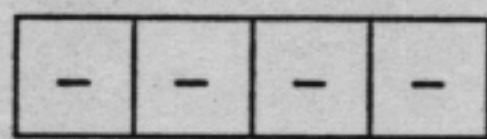
Instruction format :

CALR disp



disp = D<12:0>, but since always D<0> = 0, D<0> is not included in the instruction code.

Flag status change: z s v c



CBCD dst, src : Compare Decimal with Carry Flag

✓

Operation : dst - src - C

Description : Decimal-subtracts the contents of src and carry flag C from the contents of dst, and reflects the result in condition code flag CC.

Instruction format :

G format :

CBCD.@:G LEA,#8
CBCD.@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	0	1	1	0	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

CBCD.@:A MEM,SEA
CBCD.@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
0	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

*	-	-	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The Z flag is set to 1 when the Z flag = 1 and the operation result = 0; otherwise, cleared to zero.

MSB of sign-extended immediate data must be "0". If immediate data need to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0; Byte operation - 00~99H

Word or Double Word operation - 00~79H

A format, DD=0 and SEA=Quick Imm : 0~7

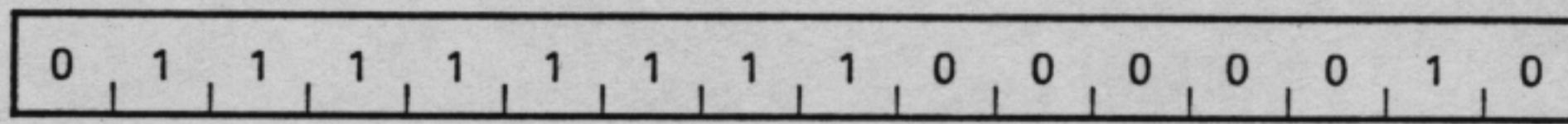
CCF : Complement of Carry Flag

Operation : $C \leftarrow \text{inverted value of } C$

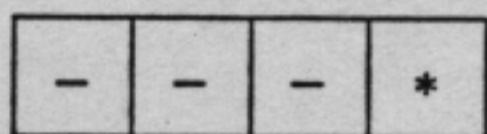
Description : Inverts the contents of carry flag C.

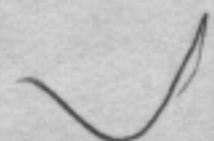
Instruction format :

CCF



Flag status change: z s v c



CHK mem, src : Check Unsigned Bound

Operation : if $\text{src} \geq (\text{mem})$ (unsigned) then if $\text{src} < (\text{mem} + \text{size})$ then $V=0, C=0$ else $V=1, C=1$ else $V=1, C=0$

Description : Checks without sign whether src is equal to or larger than (mem) and less than (mem + size).

$\text{src} < (\text{mem})$ $V=1, C=0$
$(\text{mem}) \leq \text{src} < (\text{mem} + \text{size})$ $V=0, C=0$
$(\text{mem} + \text{size}) \leq \text{src}$ $V=1, C=1$

Instruction format :

G format :

CHK.@@:G LEA, LEA

S1	S0	0	0	1	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	0	0	1	1	0	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0>$ = src, long addressing mode only

$N<7:0>$ = mem, long addressing mode only

A format :

CHK.@@:A MEM, SEA
CHK.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	0	m3	m2	m1	m0
0	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, mem is memory specified by $A<12:0>$ and src by $m<5:0>$.

When DD = 1, mem and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	*	*
---	---	---	---

-	○	○
---	---	---

Notes : size=2 :word

size=4 :double word

Register direct and immediate mode inhibited as mem addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

CHKS mem, src : Check Signed Bound

Operation : if $\text{src} \geq (\text{mem})$ (signed) then if $\text{src} < (\text{mem} + \text{size})$ then $V=0, C=0$ else $V=1, C=1$ else $V=1, C=0$

Description : Checks with sign whether src is equal to or larger than (mem) and less than (mem + size).

$\text{src} < (\text{mem})$	$V=1, C=0$
$(\text{mem}) \leq \text{src} < (\text{mem} + \text{size})$	$V=0, C=0$
$(\text{mem} + \text{size}) \leq \text{src}$	$V=1, C=1$

Instruction format :

G format :

CHKS.@:G LEA, LEA

S1	S0	0	0	1	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	0	0	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src, long addressing mode only

N<7:0> = mem, long addressing mode only

A format :

CHKS.@:A MEM, SEA
CHKS.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	0	m3	m2	m1	m0
0	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, mem is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, mem and src are exchanged.

Flag status change: z s v c

-	-	*	*
---	---	---	---

Operation size : B W D

-	○	○
---	---	---

Notes : size=2:word

size=4:double word

Register direct and immediate mode inhibited as mem addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

CLR dst : Clear Operand

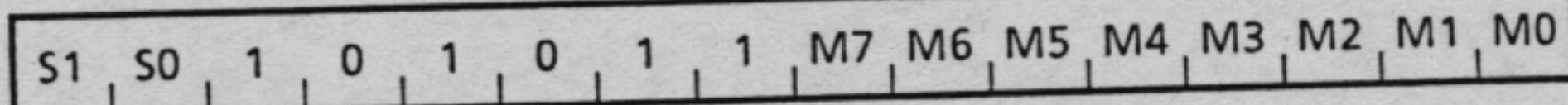
✓

Operation : $dst \leftarrow 0$

Description : Loads 0 in dst.

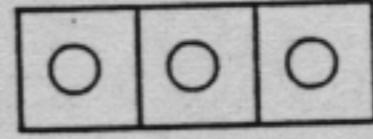
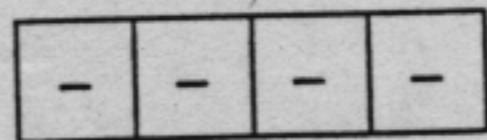
Instruction format :

CLR. @ LEA

 $M<7:0> = dst$

Flag status change: z s v c

Operation size : B W D



Notes : Immediate mode inhibited as dst addressing mode.

CP dst, src : Compare

Operation : dst - src

Description : Compares the contents of dst with src and reflects the result in condition code flag CC.

Instruction format :

S format :

CP.:@:S Reg,SEA
CP.:@:S SEA,Reg

S1	S0	0	R0	1	m5	m4	DD	R3	R2	R1	0	m3	m2	m1	m0
----	----	---	----	---	----	----	----	----	----	----	---	----	----	----	----

When DD = 0, dst is specified by R<3:0> and src by m<5:0>

When DD = 1, dst and src are exchanged.

G format :

CP.:@:G LEA,#8
CP.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	0	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

I format:

CP.:@:I LEA,#16
CP.:@:I LEA,#32

0	0	0	0	SS	0	1	0	M7	M6	M5	M4	M3	M2	M1	M0
#<15:0>															
#<31:16>															

M<7:0> = dst

A format :

CP.:@:A MEM,SEA
CP.:@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
0	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

*	*	*	*
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.
m5m4 = 11 (quick immediate) and m5m4 = 00 (register direct) in S format
allowed only when DD = 0.
Immediate mode inhibited as dst addressing mode.
m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

CPC dst, src : Compare with Carry Flag

Operation : $dst - src - C$

Description : Subtracts the contents of src and carry flag C from the contents of dst, then reflects the result in condition code flag CC.

Instruction format :

G format :

CPC.@:G LEA, #8
CPC.@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	1	1	1	0	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

CPC.@:A MEM, SEA
CPC.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
1	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$ and src by $m<5:0>$.

When DD = 1, dst and src are exchanged.

Flag status change: Z S V C Operation size : B W D

*	*	*	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The Z flag is set to 1 when the Z flag = 1 and the operation result = 0; otherwise, cleared to zero.

CPL dst : One's-Complement

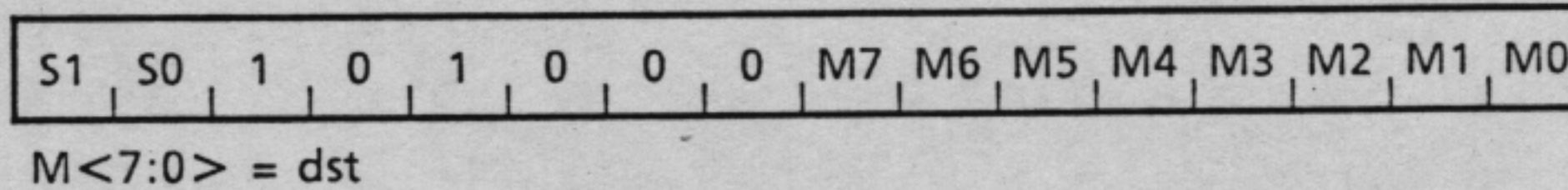
✓

Operation : $dst \leftarrow \text{one's-complement of } dst$

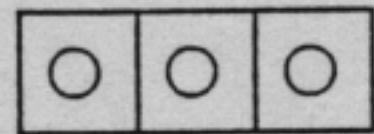
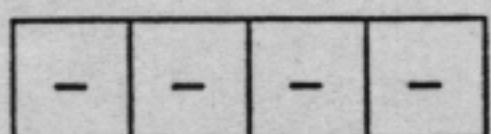
Description : Loads one's-complement of dst (0/1 inverted values of bits) in dst.

Instruction format :

CPL. @ LEA



Flag status change: z s v c Operation size : B W D



Notes : Immediate mode inhibited as dst addressing mode.

CPSN dst,src,cnts : Compare String (Unmatch detection)

Operation : $dst - src, cnts \leftarrow cnts - 1$, Repeat until $dst \neq src$ or $cnts = 0$

Description : Compares the contents of dst with src, then decrements the contents of cnts by 1. Unless $dst \neq src$ or $cnts = 0$, repeats the operation.

Instruction format :

CPSN.@ LEA,#8,Reg
CPSN.@ LEA,LEA,Reg

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	R3	R2	R1	R0	0	0	1	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

$R<3:0> = cnts$

Flag status change: z s v c Operation size : B W D

*	*	*	*
---	---	---	---

○	○	○
---	---	---

Notes : cnts always consists of words.

Flags are set according to the comparison result (dst-src).

Effective addresses of src and dst are recalculated at every repetition.

CPSZ dst, src, cnts : Compare String (Match detection)

Operation : $dst - src, cnts \leftarrow cnts - 1$, Repeat until $dst = src$ or $cnts = 0$

Description : Compares the contents of dst with src, then decrements the contents of cnts by 1. Unless $dst = src$, or $cnts = 0$, repeats the operation.

Instruction format :

CPSZ.@ LEA,#8,Reg
CPSZ.@ LEA,LEA,Reg

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	R3	R2	R1	R0	0	0	0	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)
 When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

$R<3:0> = cnts$

Flag status change: z s v c

Operation size : B W D

*	*	*	*
---	---	---	---

○	○	○
---	---	---

Notes : cnts always consists of words.

Flags are set according to the comparison result (dst-src).

Effective addresses of src and dst are recalculated at every repetition.

CSF : Complement of Sign Flag

Operation : $S \leftarrow \text{Inverted value of } S$

Description : Inverts the contents of sign flag S.

Instruction format :

CSF

0	1	1	1	1	1	1	1	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change z s v c

:

-	*	-	-
---	---	---	---

CVF: Complement of Overflow Flag

✓

Operation : $V \leftarrow \text{inverted value of } V$

Description : Inverts the contents of overflow flag V.

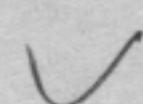
Instruction format :

CVF

0	1	1	1	1	1	1	1	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	*	-
---	---	---	---

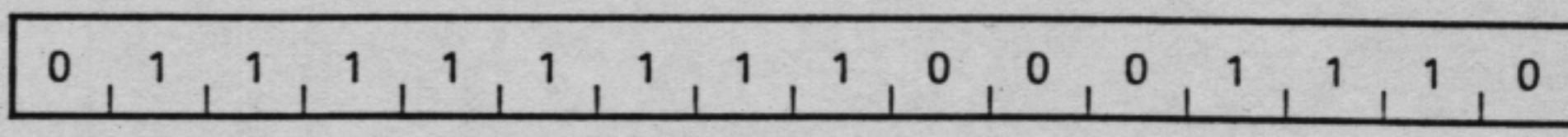
CZF : Complement of Zero Flag

Operation : $Z \leftarrow \text{inverted value of } Z$

Description : Inverts the contents of zero flag Z

Instruction format :

CZF



Flag status change: z s v c

*	-	-	-
---	---	---	---

DI : Disable Interrupt

Operation : $EI \leftarrow 0$

Description : Resets interrupt enable flag EI of the status register PSW to 0. Then only non-maskable interrupts can be accepted.

Instruction format :

DI

0	1	1	1	1	1	1	1	0	1	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

*	-	-	-
---	---	---	---

DIV dst, src : Unsigned Divide

Operation : $dst \leftarrow dst \div src$ (unsigned)

Description : Unsigned-divides the contents of dst by the contents of src, then loads the quotient in the lower half of dst and the remainder in the upper half of dst.

Instruction format :

G format :

DIV.@:G LEA,#8
DIV.@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	0	1	1	1	0	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

DIV.@:A MEM,SEA
DIV.@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	0	m3	m2	m1	m0
1	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c Operation size : B W D

-	-	*	-
---	---	---	---

O	O	-
---	---	---

Notes : Data size:

8 bit \leftarrow 16 bit \div 8 bit, remainder 8 bit : byte operation

16bit \leftarrow 32 bit \div 16 bit, remainder 16 bit : word operation

dst is read and written in double size of the operation size.

Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The V flag is set to 1 when the division result is an overflow (divided by 0 or cannot be expressed in the dst size), and the contents of dst are set to indeterminate.

MSB of sign-extended immediate data must be "0". 4 immediate data needs to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0 : Byte operation – 00~FFH

Word operation – 00~7FH

A format, DD=0 and SEA=Quick Imm : 0~7
(RW_{ntt}), (--RW_n), (RD_n++), and (--RD_n) in addressing mode are
incremented or decremented by the operation size.

DIVS dst, src : Signed Divide

Operation : $dst \leftarrow dst \div src$ (signed)

Description : Signed-divides the contents of dst by the contents of src, then loads the quotient in the lower half of dst and the remainder in the upper half of dst.

Instruction format :

G format :

DIVS.:@:G LEA,#8
DIVS.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	0	1	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

DIVS.:@:A MEM,SEA
DIVS.:@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	0	m3	m2	m1	m0
1	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$ and src by $m<5:0>$.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	*	-
---	---	---	---

O	O	-
---	---	---

Notes : Data size:

8 bit \leftarrow 16 bit \div 8 bit, remainder 8 bit: byte operation

16bit \leftarrow 32 bit \div 16 bit, remainder 16bit: word operation

dst is read and written in double size of the operation size.

Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The V flag is set to 1 when the division result is an overflow (divided by 0 or cannot be expressed in the dst size), and the contents of dst are set to indeterminate.

(RWn++), (-RWn), (RDn++), and (-RDn) in addressing mode are incremented or decremented by the operation size.

DJNZ dst, disp : Decrement and Jump Non-zero

Operation : $dst \leftarrow dst - 1$, if $dst \neq 0$ then $PC \leftarrow PC + disp$

Description : Decrements the contents of dst by 1. If the result is other than 0, the program restarts from $PC + disp$.

Instruction format :

DJNZ LEA,disp

1	SS	1	1	0	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	1	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	0

$M<7:0> = dst$

$disp = D<12:0>$, but since always = 0, $D<0>$ is not included in the instruction code.

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

-	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$disp (D<12:0>)$ is handled as signed.

DJNZC dst, cond, disp : Decrement, Conditional and Jump Non-zero

Operation : $dst \leftarrow dst - 1$, if CC is false and $dst \neq 0$ then $PC \leftarrow PC + disp$

Description : Decrements the contents of dst by 1. If the operand condition is false and the decrement result is other than zero, the program restarts execution from $PC + disp$. Otherwise, the program moves to the next instruction.

Instruction format :

DJNZC LEA,cond,disp

1	SS	1	1	0	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
C3	C2	C1	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	C0

$M<7:0> = dst$.

$disp = D<12:0>$, but since always $D<0> = 0$, $D<0>$ is not included in the instruction code.

C3 C2 C1 C0: Loop exit conditions can be written as follows:

0000 : DJNZC LEA,C,disp13 C=1 (Carry set) Notation C can also be written as ULT (Lower)

0001 : DJNZC LEA,NC,disp13 C=0 (Carry Clear) Notation NC can be also written as UGE (High or Same)

0010 : DJNZC LEA,Z,disp13 Z=1 (Equal)
 0011 : DJNZC LEA,NZ,disp13 Z=0 (Not Equal)
 0100 : DJNZC LEA,OV,disp13 V=1 (oVerflow Set)
 0101 : DJNZC LEA,NOV,disp13 V=0 (oVerflow Clear)
 0110 : DJNZC LEA,MI,disp13 S=1 (MINus)
 0111 : DJNZC LEA,PL,disp13 S=0 (PLus)
 1000 : DJNZC LEA,LE,disp13 Z+S^V=1 (Less or Equal)
 1001 : DJNZC LEA,GT,disp13 Z+S^V=0 (Greater Than)
 1010 : DJNZC LEA,LT,disp13 S^V=1 (Less Than)
 1011 : DJNZC LEA,GE,disp13 S^V=0 (Greater or Equal)
 1100 : DJNZC LEA,ULE,disp13 C+Z=1 (Low or Same)
 1101 : DJNZC LEA,UGT,disp13 C+Z=0 (HIgher)
 1111 : DJNZC LEA.A.disp13 (Always)

Flag status change: Z S V C Operation size : B W D

-	-	-	-
---	---	---	---

-	○	○
---	---	---

Notes : The loop exit condition A (Always) is insignificant, because the program unconditionally moves to the next instruction.

Immediate mode inhibited as dst addressing mode.

disp ($D<12:0>$) is handled as signed.

EI : Enable Interrupt



Operation : $EI \leftarrow 1$

Description : Sets interrupt enable flag EI of status register PSW to 1. The interrupt level is set to the level specified by IM<3:0>.

Instruction format :

EI

0	1	1	1	1	1	1	1	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	-	-
---	---	---	---

EX dst1,dst2 : Exchange

Operation : $dst1 \longleftrightarrow dst2$

Description : Exchanges the contents of dst1 and dst2.

Instruction format :

S format :

EX. @:S Reg1,Reg2

S1	S0	1	1	1	1	1	0	T3	T2	T1	T0	R3	R2	R1	R0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

T<3:0> = dst1

R<3:0> = dst2

G format :

EX. @:G LEA,LEA

S1	S0	0	0	1	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	1	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = dst2, long addressing mode only

N<7:0> = dst1, long addressing mode only

A format :

EX. @:A MEM,SEA

EX. @:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
1	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst1 is memory specified by A<12:0>
and dst2 by m<5:0>.

When DD = 1, dst1 and dst2 are exchanged.

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst1 or dst2 addressing mode.

Operation sequence which dst is CBP is different depending on the operation size and the location of CBP.

In Byte or Word operation :

- dst1=CBP, dst2=Reg - ① Write Reg.
 ② Write CBP
 ③ Exchange Bank
- dst1=Reg, dst2=CBP - ① Write CBP
 ② Exchange Bank
 ③ Write Reg. which is the exchanged Reg.

In Double word operation :

- $dst1 = CBP, dst2 = Reg$ – ① Write lower word Reg.
 ② Write CBP
 ③ Exchange Bank
 ④ Write upper word Reg which is the exchanged Reg.
- $dst1 = Reg, dst2 = CBP$ – ① Write CBP
 ② Exchange Bank
 ③ Write Reg which is the exchanged Reg.

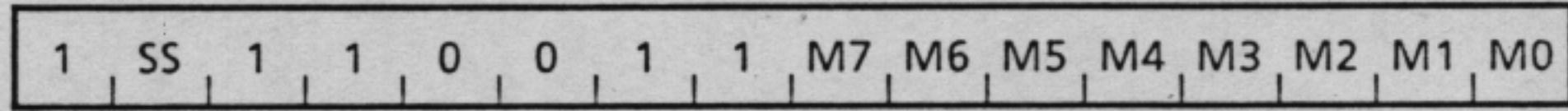
EXTS dst : Sign Extend

Operation : All bits of dst<upper> \leftarrow signed bits of dst<lower>

Description : Loads the sign bit of dst lower half to all bits of dst upper half.

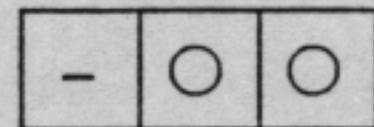
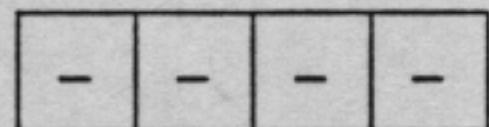
Instruction format :

EXTS.@ LEA



M<7:0> = dst.

Flag status change: z s v c Operation size : B W D



Notes : Immediate mode inhibited as dst addressing mode.

dst is read in the operation size, sign-extended using the lower half of dst, then rewritten in the operation size.

EXTZ dst : Zero Extend

Operation : All bits of dst<upper> $\leftarrow 0$

Description : Clears the upper half of dst.

Instruction format :

EXTZ.0 LEA

1	SS	1	1	0	0	1	0	M7	M6	M5	M4	M3	M2	M1	M0
---	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

M<7:0> = dst.

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

-	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

dst is read in the operation size, zero-extended for the upper half, then rewritten in the operation size.

HALT : Halt CPU

✓

Operation : Halt CPU

Description : Halts instruction execution. Receiving an interrupt restarts instruction execution.

Instruction format :

HALT

0	1	1	1	1	1	1	1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	-	-
---	---	---	---

JP dst : Jump

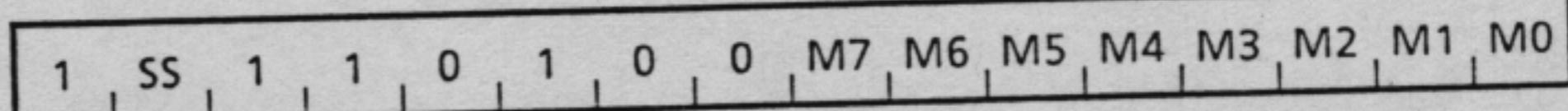
Operation : $PC \leftarrow \text{effective address of dst}$

✓

Description : Jumps to the effective address of dst.

Instruction format :

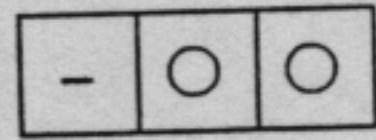
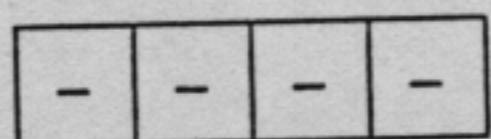
JP. @ LEA



$M<7:0> = \text{dst}$

Flag status change: z s v c

Operation size : B W D



Notes : Register direct or immediate mode inhibited as dst addressing mode.
The operation size is employed to increment or decrement an address by
(RWn++), (--RWn), (RDn++) or (--RDn) in addressing mode.
When the effective address of dst is odd numbered, jumps to the even
numbered address obtained by rounding off the least significant bit to 0.

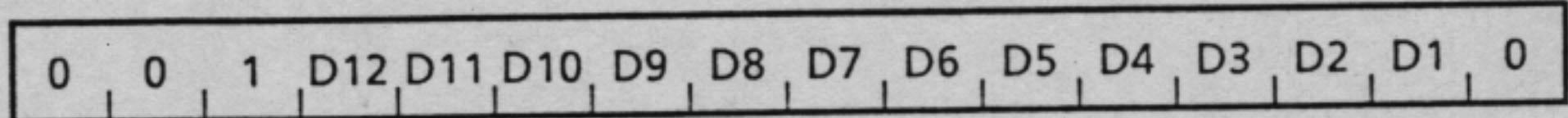
JR disp: Relative Jump

Operation : $PC \leftarrow PC + \text{disp}$

Description : Restarts program execution from $PC + \text{disp}$.

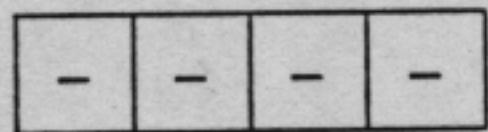
Instruction format :

JR disp



disp = $D_{12:0}$, but since always $D_{0:0} = 0$, $D_{0:0}$ is not included in the instruction code.

Flag status change: z s v c



Notes : disp ($D_{12:0}$) is handled as signed.

JRBC num,abs,disp : Bit Test Relative Jump If Clear

V

Operation : if $abs < num > = 0$ then $PC \leftarrow PC + disp$

Description : If bit num at absolute address abs is cleared to zero, program execution restarts from $PC + disp$.

Instruction format :

JRBC num, MEM, disp

0	0	0	1	1	1	1	D8	D7	D6	D5	D4	D3	D2	D1	0
B2	B1	B0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

disp = $D < 8:0 >$, but since always $D < 0 > = 0$, $D < 0 >$ is not included in the instruction code.

abs is memory specified by $A < 12:0 >$.

num is bit number specified by $B < 2:0 >$.

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	-	-
---	---	---

Notes : disp ($D < 8:0 >$) is handled as signed.

JRBS num,abs,disp : Bit Test Relative Jump If Set

Operation : if $\text{abs} < \text{num} > = 1$ then $\text{PC} \leftarrow \text{PC} + \text{disp}$

Description : If bit num at absolute address abs is set to one, program execution restarts from $\text{PC} + \text{disp}$.

Instruction format :

JRBS num, MEM, disp

0	0	0	1	1	1	1	D8	D7	D6	D5	D4	D3	D2	D1	1
B2	B1	B0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

disp = $D < 8:0 >$, but since always $D < 0 > = 0$, $D < 0 >$ is not included in the instruction code.

abs is memory specified by $A < 12:0 >$.

num is bit number specified by $B < 2:0 >$.

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	-	-
---	---	---

Notes : disp ($D < 8:0 >$) is handled as signed.

JRC cond, disp : Relative Jump Conditional

✓

Operation : If CC is true, then $PC \leftarrow PC + \text{disp}$

Description : If the operand condition is true, program execution restarts from $PC + \text{disp}$.

Instruction format :

JRC cond, disp

0	0	0	1	C3	C2	C1	D8	D7	D6	D5	D4	D3	D2	D1	C0
---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----

disp = $D<8:0>$, but since always $D<0> = 0$, $D<0>$ is not included in the instruction code.

C3 C2 C1 C0: Jump conditions can be written as follows:

0000 : JRC C, disp9 C=1 (Carry set) Notation C can also be written as ULT (Lower)

0001 : JRC NC, disp9 C=0 (Carry Clear) Notation NC can also be written as UGE (High or Same)

0010 : JRC Z, disp9 Z=1 (EQual)

0011 : JRC NZ, disp9 Z=0 (Not Equal)

0100 : JRC OV, disp9 V=1 (oVerflow Set)

0101 : JRC NOV, disp9 V=0 (oVerflow Clear)

0110 : JRC MI, disp9 S=1 (MINus)

0111 : JRC PL, disp9 S=0 (PLus)

1000 : JRC LE, disp9 Z+S^V=1 (Less or Equal)

1001 : JRC GT, disp9 Z+S^V=0 (Greater Than)

1010 : JRC LT, disp9 S^V=1 (Less Than)

1011 : JRC GE, disp9 S^V=0 (Greater or Equal)

1100 : JRC ULE, disp9 C+Z=1 (Low or Same)

1101 : JRC UGT, disp9 C+Z=0 (HIgher)

Flag status change: z s v c

-	-	-	-
---	---	---	---

Notes : disp ($D<8:0>$) is handled as signed.

LD dst,src : Load

Operation : $dst \leftarrow src$

Description : Loads the contents of src in dst.

Instruction format :

S format :

LD.@@:S Reg,SEA
LD.@@:S SEA,Reg

S1	S0	0	R0	1	m5	m4	DD	R3	R2	R1	1	m3	m2	m1	m0
----	----	---	----	---	----	----	----	----	----	----	---	----	----	----	----

When DD = 0, dst is specified by R<3:0>; src, by m<5:0>.

When DD = 1, dst and src are exchanged.

G format :

LD.@@:G LEA,#8
LD.@@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	0	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

I format :

LD.@@:I LEA,#16
LD.@@:I LEA,#32

0	0	0	0	SS	0	1	1	M7	M6	M5	M4	M3	M2	M1	M0
#<15:0>															
#<31:16>															

M<7:0> = dst

A format :

LD.@@:A MEM,SEA
LD.@@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
0	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>; src, by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.
m5m4 = 11 (quick immediate) and m5m4 = 00 (register direct) in S format
allowed only when DD = 0.
Immediate mode inhibited as dst addressing mode.
m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

LDA dst, src : Load Effective Address

Operation : $dst \leftarrow \text{effective address of src}$

Description : Loads the effective address of src in dst.

Instruction format :

N M

LDA.	@	LEA,LEA	1	SS	1	1	0	0	0	0	M7	M6	M5	M4	M3	M2	M1	M0
			1	0	0	1	0	1	1	1	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = \text{src, long addressing mode only}$
 $N<7:0> = \text{dst, long addressing mode only}$

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

-	O	O
---	---	---

Notes : Register direct or immediate mode inhibited as src addressing mode.

LDCF src, num : Load Bit to Carry Flag

✓

Operation : $C \leftarrow \text{src} < \text{num} >$

Description : Loads the contents of bit num in src in carry flag C.

Instruction format :

G format :

LDCF.@@:G LEA, #8
LDCF.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	1	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = src, long addressing mode only

A format :

LDCF.@@:A MEM, SEA
LDCF.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
1	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, src is specified by A<12:0> and num by m<5:0>.

When DD = 1, num and src are exchanged.

Flag status change: z s v c

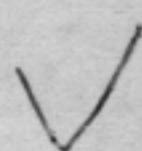
Operation size : B W D

-	-	-	*
---	---	---	---

○	○	○
---	---	---

Notes : m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4 bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

LDS dst, src, cnts : Load String

Operation : $dst \leftarrow src, cnts \leftarrow cnts - 1$, Repeat until $cnts = 0$

Description : Loads the contents of src in dst, then decrements the contents of cnts by
1. If the decrement result is other than 0, the operation is repeated.

Instruction format :

LDS.@ LEA,#8,Reg
LDS.@ LEA,LEA,Reg

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	R3	R2	R1	R0	0	1	1	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0>$ = src When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode

$N<7:0>$ = dst, long addressing mode only

$R<3:0>$ = cnts

Flag status change: z s v c

-	-	-	-
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : cnts always consists of words.

Effective addresses of src and dst are recalculated at every repetition.

Immediate and special register (Imm, SP, ISP, PBP, CBP, PSW, IMC, CC)
mode inhibited as dst addressing mode.

LINK disp : Link Stack Frame

Operation : $(-SP) \leftarrow R2, R2 \leftarrow SP, SP \leftarrow SP + disp$

Description : Saves the contents of R2 in the stack area, then loads the contents of stack pointer SP in R2, adds the contents of the stack pointer and signed disp, and loads the result in the stack pointer.

Instruction format :

LINK disp

1	1	0	0	0	0	0	D8	D7	D6	D5	D4	D3	D2	D1	1
---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	---

Flag status change: z s v c

-	-	-	-
---	---	---	---

Notes : From the point of view of execution speed, it is recommended that the stack pointer be an even number. Therefore, the least significant bit of D<8:0> is always 0.

U

MAC dst, src1, src2 : Unsigned Multiply-and-Add Calculation

Operation : $dst \leftarrow src1 \times src2 + dst$

Description : Unsigned-multiplies the contents of src1 by the contents of src2, adds the contents of dst to the result, and loads the result in dst.

Instruction format :

MAC. @	Reg, LEA, #8	S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
MAC. @	Reg, LEA, LEA	0	R3	R2	R1	R0	0	1	0	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src2$ When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode

$N<7:0> = src1$, long addressing mode only

$R<3:0> = dst$

Flag status change: z s v c Operation size : B W D

*	*	*	*
---	---	---	---

○	○	-
---	---	---

Notes : Data size:

16bit \leftarrow 16bit + 8bit \times 8bit : byte operation

32bit \leftarrow 32bit + 16bit \times 16bit : word operation

dst is read and written in double size of the operation size.

The flag status changes according to the operation result: $(src1 \times src2) + dst$.

Immediate mode inhibited as src1 addressing mode.

MSB of sign-extended immediate data must be "0". If immediate data needs to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD

II=0: Byte operation - 00~FFH

Word operation - 00~7FH

MACS dst,src1,src2 : Signed Multiply-and-Add Calculation

✓

Operation : $dst \leftarrow src1 \times src2 + dst$

Description : Signed-multiplies the contents of src1 by the contents of src2, adds the contents of dst to the result, and loads the result in dst.

Instruction format :

MACS.@ Reg,LEA,#8
MACS.@ Reg,LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
0	R3	R2	R1	R0	0	1	1	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = src2$ When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode

$N<7:0> = src1$, long addressing mode only

$R<3:0> = dst$

Flag status change: z s v c

*	*	*	*
---	---	---	---

Operation size : B W D

○	○	-
---	---	---

Notes : Data size :

16bit \leftarrow 16bit + 8bit \times 8bit : byte operation

32bit \leftarrow 32bit + 16bit \times 16bit : word operation

dst is read and written in double size of the operation size.

The flag status changes according to the operation result: $(src1 \times src2) + dst$.

Immediate mode inhibited as src1 addressing mode.

MAX dst, src : Unsigned Maximum Value

V

Operation : if $dst \geq src$ (unsigned) then $dst \leftarrow dst$ else $dst \leftarrow src$

Description : Unsigned-compares the contents of dst with the contents of src. If the contents of dst are equal to or larger than those of src, they are loaded in dst; if smaller, the contents of src are loaded in dst.

Instruction format :

G format :

MAX. @:G LEA, #8
MAX. @:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	1	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src$ When $II = 0$, immediate data (8 bits, signed)

When $II = 1$, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

MAX. @:A MEM, SEA
MAX. @:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
1	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When $DD = 0$, dst is memory specified by $A<12:0>$; src, by $m<5:0>$.

When $DD = 1$, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

*	*	*	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when $DD = 0$.

The flag changes according to the dst-src result.

MSB of sign-extended immediate data must be "0". If immediate data needs to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and $II = 0$: Byte operation - 00~FFH

Word operation - 00~7FH

A format, $DD = 0$ and $SEA = \text{Quick Imm}$: 0~7

MAXS dst, src : Signed Maximum Value

✓

Operation : if $dst \geq src$ (signed) then $dst \leftarrow dst$ else $dst \leftarrow src$

Description : Signed-compares the contents of dst with the contents of src. If the contents of dst are equal to or larger than those of src, they are loaded in dst; if smaller, the contents of src are loaded in dst.

Instruction format :

G format :

MAXS.@@:G LEA, #8
MAXS.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	1	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src$ When $II = 0$, immediate data (8 bits, signed)

When $II = 1$, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

MAXS.@@:A MEM, SEA
MAXS.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
1	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When $DD = 0$, dst is memory specified by $A<12:0>$ and src by $m<5:0>$.

When $DD = 1$, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

*	*	*	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when $DD = 0$.

The flag changes according to the dst-src result.

MIN dst, src : Unsigned Minimum Value ✓

Operation : if $dst \leq src$ (unsigned) then $dst \leftarrow dst$ else $dst \leftarrow src$

Description : Unsigned-compares the contents of dst with the contents of src. If the contents of dst are equal to or smaller than those of src, they are loaded in dst; if larger, the contents of src are loaded in dst.

Instruction format :

G format :

MIN. @:G LEA, #8
MIN. @:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	1	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N<7:0> = dst$, long addressing mode only

A format :

MIN. @:A MEM, SEA
MIN. @:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
1	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$; src, by $m<5:0>$.

When DD = 1, dst and src are exchanged.

Flag status change:

*	*	*	*
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The flag changes according to the dst-src result.

MSB of sign-extended immediate data must be "0". If immediate data needs to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0: Byte operation - 00~FFH

Word operation - 00~7FH

A format, DD=0 and SEA=Quick Imm : 0~7

MINS dst, src : Signed Minimum Value

J

Operation : if $dst \leq src$ (signed) then $dst \leftarrow dst$ else $dst \leftarrow src$

Description : Signed-compares the contents of dst with the contents of src. If the contents of dst are equal to or smaller than those of src, they are loaded in dst; if larger, the contents of src are loaded in dst.

Instruction format :

G format :

MINS.@@:G LEA, #8
MINS.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	1	1	0	1	N7	N6	N5	N4	N3	N2	N1	NO

$M<7:0> = src$ When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode
 $N<7:0> = dst$, long addressing mode only

A format :

MINS.@@:A MEM, SEA
MINS.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
1	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A<12:0>$; src, by $m<5:0>$.
When DD = 1, dst and src are exchanged.

Flag status change: z s v c

*	*	*	*
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The flag changes according to the dst-src result.

MIRR dst : Mirror Exchange

Operation : dst<MSB:LSB> \leftarrow dst<LSB:MSB>

Description : Mirror-exchanges the contents of dst as the bit pattern image.

Instruction format :

MIRR. @ LEA

S1	S0	1	0	0	0	1	1	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

M<7:0> = dst

Flag status change: z s v c Operation size : b w d

-	-	-	-
---	---	---	---

o	o	o
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

MUL dst, src : Unsigned Multiply

Operation : $dst \leftarrow dst \times src$ (unsigned)

Description : Unsigned-multiplies the contents of dst by those of src, then loads the result in dst.

Instruction format :

G format :

MUL. @:G LEA, #8
MUL. @:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	0	1	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

MUL. @:A MEM, SEA
MUL. @:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	0	m3	m2	m1	m0
1	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>; src, by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change:

-	-	-	-
---	---	---	---

Operation size : B W D

○	○	-
---	---	---

Notes : Data size:

16 bit \leftarrow 8 bit \times 8bit : byte operation

32 bit \leftarrow 16 bit \times 16bit: word operation

dst is read and written in double size of the operation size.

Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

MSB of sign-extended immediate data must be "0". If immediate data needs to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0 :Byte operation - 00~FFH

Word operation - 00~7FH

A format, DD=0 and SEA=Quick Imm : 0~7

(RWn++), (--RWn), (RDn++), and (--RDn) in addressing mode are incremented or decremented by operation size.

MULS dst, src : Signed Mutiply

Operation : $dst \leftarrow dst \times src$ (signed)

Description : Signed-multiplies the contents of dst by those of src, then loads the result in dst.

Instruction format :

G format :

MULS.@@:G LEA,#8
MULS.@@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	0	1	1	0	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

MULS.@@:A MEM,SEA
MULS.@@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	0	m3	m2	m1	m0
1	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>; src, by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

○	○	-
---	---	---

Notes : Data size:

16 bit \leftarrow 8 bit \times 8bit : byte operation

32 bit \leftarrow 16 bit \times 16bit: word operation

dst is read and writen in double size of the operation size.

Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

(RWn++), (--RWn), (RDn++), and (--RDn) in addressing mode are incremented or decremented by operation size.

NEG dst : Negate

Operation : $dst \leftarrow 0 - dst$

Description : Subtracts the contents of dst from 0, then loads the result in dst.

Instruction format :

NEG. @ LEA

S1	S0	1	0	1	0	0	1	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

$M<7:0> = dst$

Flag status change: z s v c Operation size : B W D

*	*	*	*
---	---	---	---

○	○	○
---	---	---

Notes : Flags are set according to the operation result.

Immediate mode inhibited as dst addressing mode.

NOP : No Operation

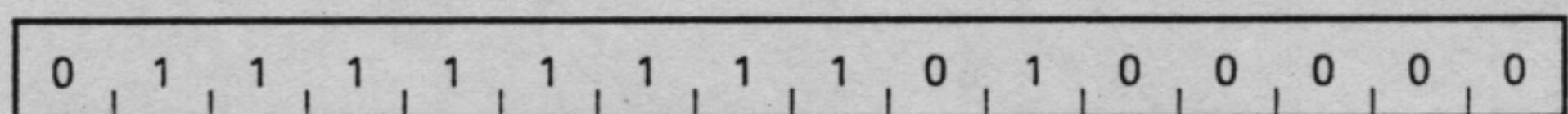
✓

Operation : No operation

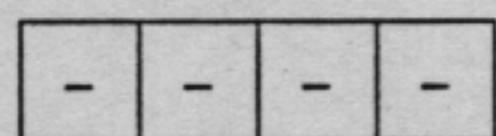
Description : No operation, moves to the next instruction.

Instruction format :

NOP



Flag status change: z s v c



OR dst, src : Logical OR

J

Operation : $dst \leftarrow dst \text{ OR } src$

Description : ORs the contents of dst and src, then loads the result in dst.

Instruction format :

S format :

OR.@@:S Reg1,Reg2

S1	S0	1	1	1	0	1	0	T3	T2	T1	T0	R3	R2	R1	RO
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

T<3:0> = dst

R<3:0> = src

G format :

OR.@@:G LEA,#8
OR.@@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	0	1	0	1	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

I format:

OR.@@:I LEA,#16
OR.@@:I LEA,#32

0	0	0	0	SS	1	0	1	M7	M6	M5	M4	M3	M2	M1	M0
#<15:0>															
#<31:16>															

M<7:0> = dst

A format :

OR.@@:A MEM,SEA
OR.@@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
0	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>; src, by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change:

*	*	0	0
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.
Immediate mode inhibited as dst addressing mode.
m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

ORCF src, num : Bit Logical OR with Carry Flag

✓

Operation : $C \leftarrow C \text{ OR } \text{src} < \text{num} >$

Description : ORs the contents of carry flag C with bit num in src, then loads the result in C.

Instruction format :

G format :

ORCF.@@:G LEA, #8
ORCF.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	1	1	0	1	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = num. When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = src, long addressing mode only

A format :

ORCF.@@:A MEM, SEA
ORCF.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
1	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, src is memory specified by A<12:0>

and num is number specified by m<5:0>.

When DD = 1, num and src are exchanged.

Flag status change: z s v c

Operation size : B W D

-	-	-	*
---	---	---	---

○	○	○
---	---	---

Notes : m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

POP dst : Pop from Stack

✓

Operation : $dst \leftarrow (SP +)$

When byte : $dst \leftarrow (SP)$, $SP \leftarrow SP + 1$

When word : $dst \leftarrow (SP)$, $SP \leftarrow SP + 2$

When double word : $dst \leftarrow (SP)$, $SP \leftarrow SP + 4$

Description : Loads the contents at the memory address specified by the stack pointer SP in dst, then adds the number of bytes in the operand to the stack pointer.

Instruction format :

POP. @ LEA

S1	S0	1	0	0	0	0	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	----	----	----	----	----	----	----	----

$M<7:0> = dst$

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : The data is read from the stack area after the effective address of dst is calculated.

Immediate mode inhibited as dst addressing mode.

PUSH src : Push to Stack

Operation : $(-SP) \leftarrow src$

When byte : $SP \leftarrow SP - 1, (SP) \leftarrow src$

When word : $SP \leftarrow SP - 2, (SP) \leftarrow src$

When double word : $SP \leftarrow SP - 4, (SP) \leftarrow src$

Description : Subtracts the number of bytes in the operand from stack pointer SP, then loads the contents of src in the memory address specified by the stack pointer.

Instruction format :

PUSH.@ LEA

S1	S0	1	0	0	0	0	1	M7	M6	M5	M4	M3	M2	M1	M0
M<7:0> = src															

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : The data is written to the stack area after src is read.

PUSHA dst : Push Effective Address to Stack

V

Operation : $(-SP) \leftarrow$ effective address of dst

When word : $SP \leftarrow SP - 2, (SP) \leftarrow$ effective address of dst

When double word : $SP \leftarrow SP - 4, (SP) \leftarrow$ effective address of dst

Description : Pushes the effective address of dst to the stack area.

Instruction format :

PUSHA.@ LEA

1	SS	1	1	0	0	0	1	M7	M6	M5	M4	M3	M2	M1	M0
---	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

M<7:0> = dst

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

-	○	○
---	---	---

Notes : The data is written to the stack area after the effective address of dst is calculated.

Register direct or immediate mode inhibited as dst addressing mode.

RCF : Reset Carry Flag

Operation : $C \leftarrow 0$

Description : Resets carry flag C to 0.

Instruction format :

RCF

0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	-	0
---	---	---	---

RET : Return ✓

Operation : $PC \leftarrow (SP), SP \leftarrow SP + 4$

Description : Pops the return address from the stack area to program counter PC, then starts program execution from the program address of the restored PC.

Instruction format :

RET

0	1	1	1	1	1	1	1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	-	-
---	---	---	---

RETD disp : Return And Delete Parameter Area

Operation : $PC \leftarrow (SP), SP \leftarrow SP + 4, SP \leftarrow SP + \text{disp}$

Description : Pops the return address from the stack area to program counter PC, then adds the disp (signed) value to stack pointer SP.

Instruction format :

RETD disp

1	1	0	0	1	0	0	D8	D7	D6	D5	D4	D3	D2	D1	1
---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	---

Flag status change: z s v c

-	-	-	-
---	---	---	---

Notes : From the point of view of execution speed, it is recommended that the stack pointer be an even number. Therefore, the least significant bit of D<8:0> is always 0.

✓

RETI : Return from Interrupt

✓

Operation : Returns special registers and general-purpose registers from the memory stack area or bank.

Description : Returns special registers and general-purpose registers from the memory stack area or bank according to the contents of bit RA in PSW. Starts program execution from the program address of the restored program counter.

Instruction format :

RETI

0	1	1	1	1	1	1	1	0	1	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

*	*	*	*
---	---	---	---

Notes : If memory stack, only PC and PSW are restored.
If bank, PC, PBP, CBP, and PSW are restored.
Flags are set (restored) according to the restored PSW contents.
It is used for returns from general interrupt processing routines.

RETS : Return from Single Step Interrupt

Operation : Restores special registers and general-purpose registers from the memory stack area or bank.

Description : Restores special registers and general-purpose registers from the memory stack memory area or bank according to the contents of bit RA in PSW. Starts program execution from the program address of the restored program counter.

Instruction format :

RETS

0	1	1	1	1	1	1	1	0	1	0	1	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

*	*	*	*
---	---	---	---

Notes : If memory stack, only PC and PSW are restored.

If bank, PC, PBP, CBP, and PSW are restored.

Flags are set (restored) according to the restored PSW contents.

Unlike RETI, no interrupt can be accepted until execution of the next instruction is completed. Used as a return instruction from a single step interrupt. Ensures implementation of a single step interrupt for every instruction.

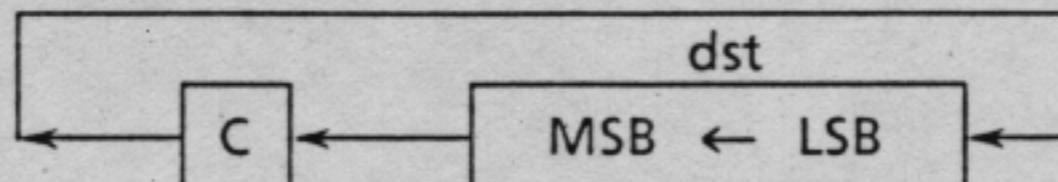
RL dst, num : Rotate Left with Carry Flag

V

Operation : C & dst \leftarrow rotate left C & dst, repeats by the number of num

Description : Rotates left the result of linking the carry flag to the upper bits of dst. Repeats this operation by the number of num. The number of the rotation in S format is once only.

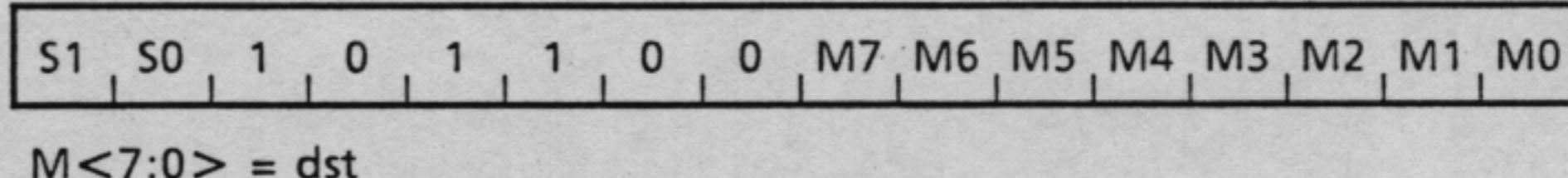
Description figure :



Instruction format :

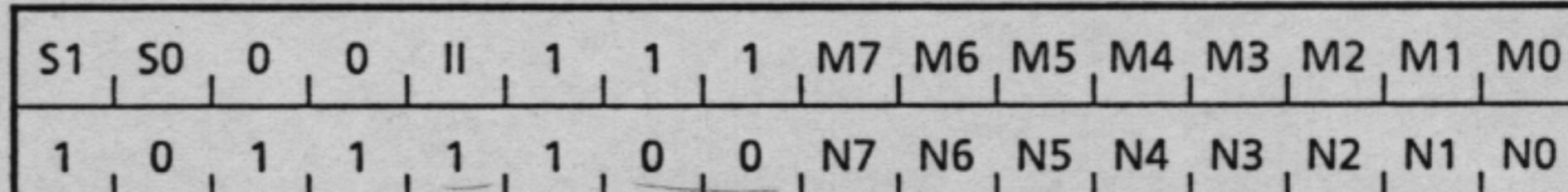
S format :

RL.:@:S LEA,1



G format :

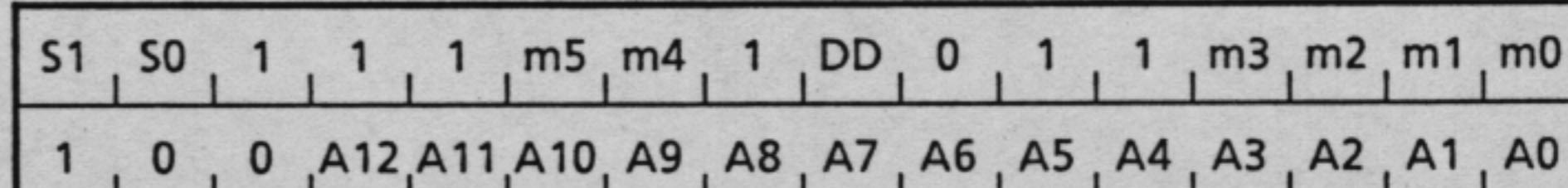
RL.:@:G LEA,#8
RL.:@:G LEA,LEA



M<7:0> = num When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode
N<7:0> = dst, long addressing mode only

A format :

RL.:@:A MEM,SEA
RL.:@:A SEA,MEM



When DD = 0, dst is memory specified by A<12:0>
and num is number specified by m<5:0>.
When DD = 1, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D

*	*	0	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

When the number of bits to be shifted is 0, dst and the C flag do not change.

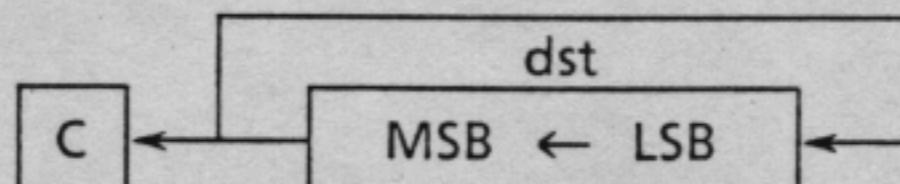
The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

RLC dst, num : Rotate Left without Carry Flag

Operation : $C \leftarrow dst < MSB >$, $dst \leftarrow \text{rotate left } dst$,
 $dst < LSB > \leftarrow dst < MSB >$, repeats by the number of num

Description : Loads the contents of MSB of dst in carry flag C, rotates left the contents of dst, and loads the MSB of dst in LSB . Repeats this operation by the number of num. The number of the rotation in S format is once only.

Description figure :



Instruction format :

S format :

RLC.@:S LEA,1

S1	S0	1	0	1	1	1	0	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

$M < 7:0 > = dst$

G format :

RLC.@:G LEA,#8
RLC.@:G LEA,LEA

S1	S0	0	0		1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	1	1	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

$M < 7:0 > = num$ When $|| = 0$, immediate data (8 bits, signed)
When $|| = 1$, long addressing mode

$N < 7:0 > = dst$, long addressing mode only

A format :

RLC.@:A MEM,SEA
RLC.@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	1	m3	m2	m1	m0
1	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When $DD = 0$, dst is memory specified by $A < 12:0 >$
and num is number specified by $m < 5:0 >$.
When $DD = 1$, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D

*	*	0	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when $DD = 0$.

When the number of bits to be shifted is 0, dst and the C flag do not change.

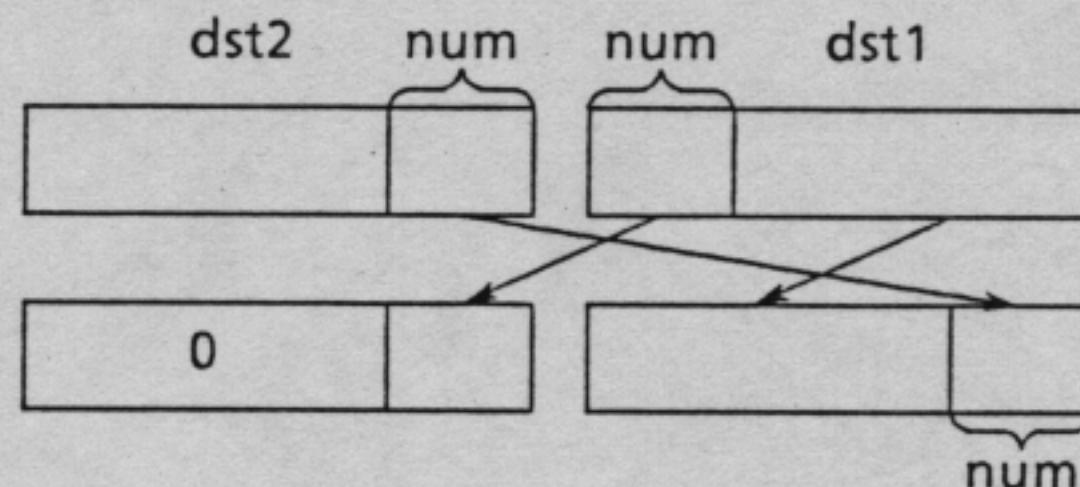
The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

RLM dst1, dst2, num : Rotate Left Multi Bit

Operation : Lower num bits in dst2&dst1 \leftarrow rotate left lower num bits in dst2&dst1, repeats by the number of num.

Description : Links the contents of lower num bits in dst2 with dst1, then rotates left by the num bits.

Description figure :



Instruction format :

RLM.0 LEA,Reg,#8
RLM.0 LEA,Reg,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
0	R3	R2	R1	RO	1	0	0	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst1, long addressing mode only

R<3:0> = dst2

Flag status change: z s v c

Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst1 addressing mode.

dst2 always consists of words. The upper bits not used are cleared to zero.

The lower 3 bits of num are effective when the operation size is byte; the lower 4 bits, when the operation size is word/double word.

When the number of bits to be shifted is 0, dst1 does not change, and dst2 is cleared to zero.

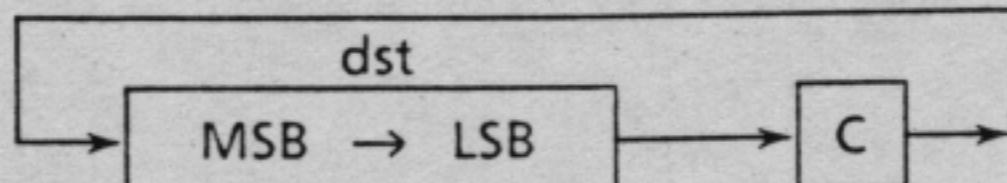
operation result write in dst2 and then in dst1.

RR dst, num : Rotate Right with Carry Flag

Operation : $C \& dst \leftarrow \text{rotate right } C \& dst$, repeats by the number of num

Description : Rotates right the result of linking the carry flag to the upper bits of dst. Repeats this operation by the number of num. The number of the rotation in S format is once only.

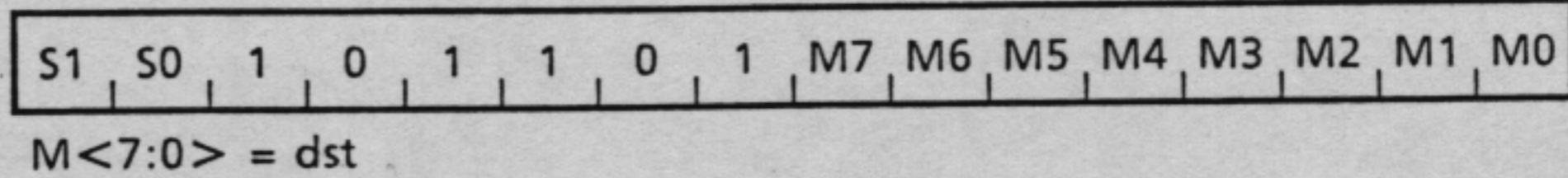
Description figure :



Instruction format:

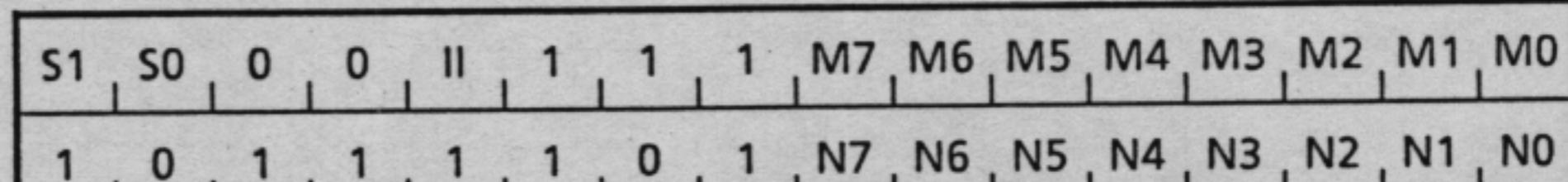
S format :

RR.:@:S LEA,1



G format :

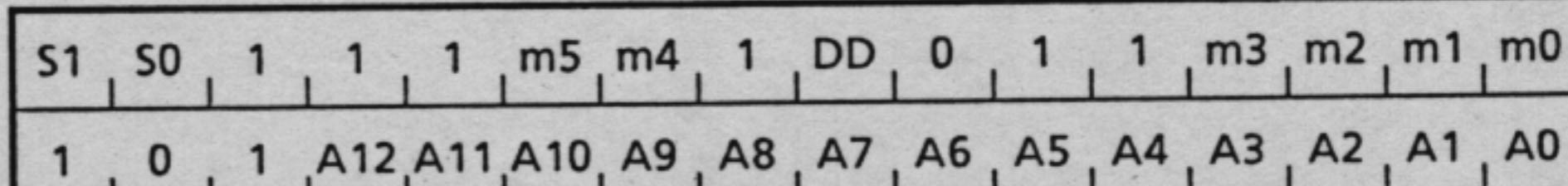
RR.:@:G LEA,#8
RR.:@:G LEA,LEA



M<7:0> = num When || = 0, immediate data (8 bits, signed)
When || = 1, long addressing mode
N<7:0> = dst, long addressing mode only

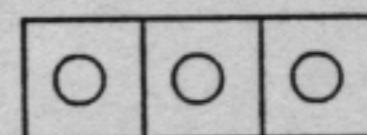
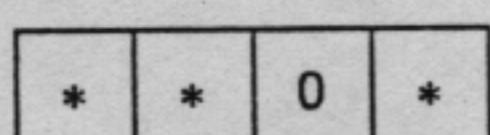
A format :

RR.:@:A MEM,SEA
RR.:@:A SEA,MEM



When DD = 0, dst is memory specified by A<12:0>
and num is number specified by m<5:0>.
When DD = 1, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D



Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

When the number of bits to be shifted is 0, dst and the C flag do not change.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4 bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

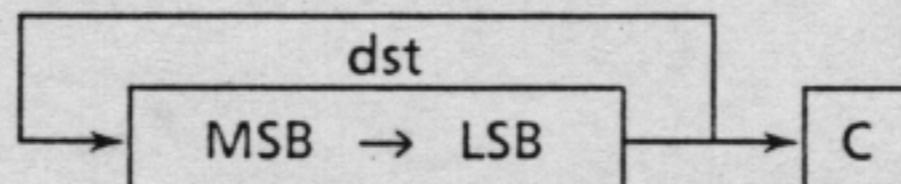
RRC dst, num : Rotate Right without Carry Flag

✓

Operation : $C \leftarrow dst < LSB >$, $dst \leftarrow$ rotate right value of dst ,
 $dst < MSB > \leftarrow dst < LSB >$, repeats by the number of num

Description : Loads the contents of LSB of dst in carry flag C , rotates right the contents of dst , and loads the LSB of dst in MSB. Repeats this operation by the number of num . The number of the rotation in S format is once only.

Description figure :



Instruction format :

S format :

RRC. @:S LEA, 1

S1	S0	1	0	1	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

$M < 7:0 > = dst$

G format :

RRC. @:G LEA, #8
RRC. @:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	1	1	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

$M < 7:0 > = num$ When $II = 0$, immediate data (8 bits, signed)
 When $II = 1$, long addressing mode

$N < 7:0 > = dst$, long addressing mode only

A format :

RRC. @:A MEM, SEA
RRC. @:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	1	m3	m2	m1	m0
1	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When $DD = 0$, dst is memory specified by $A < 12:0 >$

and num is number specified by $m < 5:0 >$.

When $DD = 1$, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D

*	*	0	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when $DD = 0$.

When the number of bits to be shifted is 0, dst and the C flag do not change.

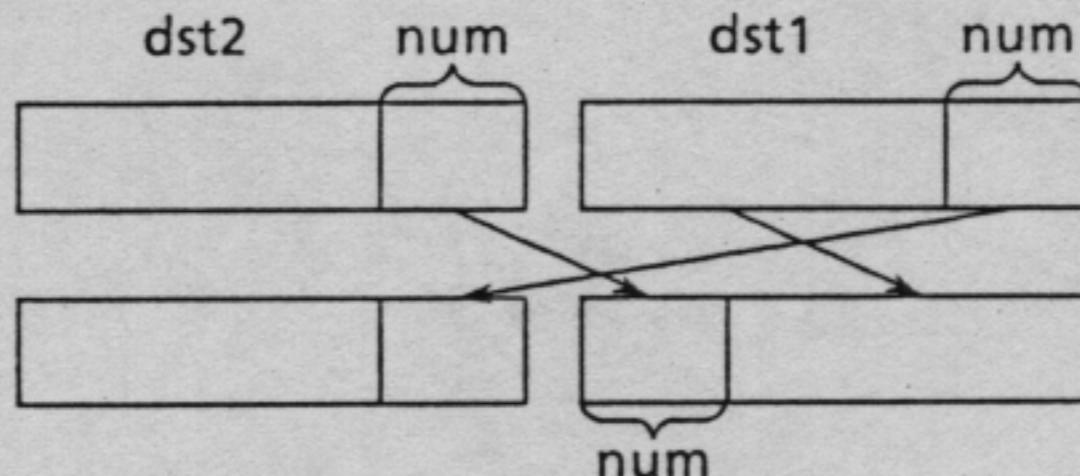
The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4 bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

RRM dst1, dst2, num : Rotate Right Multi Bit

Operation : Lower num bits of dst2&dst1 \leftarrow rotate right lower num bits of dst2 & dst1, repeats by the number of num.

Description : Links the contents of lower num bits of dst2 and dst1, then rotates right by the num bits.

Description figure :



Instruction format :

RRM. @ LEA, Reg, #8
RRM. @ LEA, Reg, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
0	R3	R2	R1	RO	1	0	1	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = num When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst1, long addressing mode only

R<3:0> = dst2

Flag status change: z s v c

-	-	-	-
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst1 addressing mode.

dst2 always consists of words. The upper bits not used are cleared to zero.

The lower 3 bits of num are effective when the operation size is byte; the lower 4 bits, when the operation size is word/double word.

When the number of bits to be shifted is 0, dst1 does not change, and dst2 is cleared to zero.

operation results write in dst2 and then in dst1.

RSF : Reset Sign Flag ✓

Operation : $S \leftarrow 0$

Description : Resets the sign flag S to 0.

Instruction format :

RSF

0	1	1	1	1	1	1	1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	0	-	-
---	---	---	---

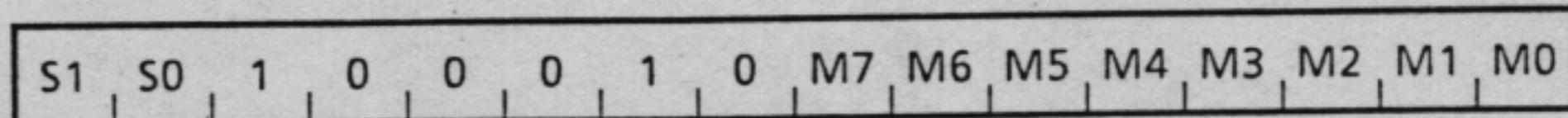
RVBY dst : Reverse Byte

Operation : $dst \leftarrow \text{reverse byte order of } dst$

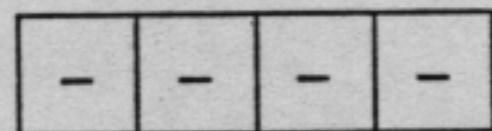
Description : Reverses the byte order of dst.

Instruction format :

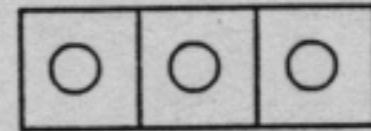
RVBY.@ LEA



Flag status change: z s v c



Operation size : B W D



Notes : Immediate mode inhibited as dst addressing mode.

When the operation size is byte, the contents of dst do not change.

RVF : Reset Overflow Flag

Operation : $V \leftarrow 0$

Description : Resets the overflow flag V to 0.

Instruction format :

RVF

0	1	1	1	1	1	1	1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	0	-
---	---	---	---

RZF : Reset Zero Flag

Operation : $Z \leftarrow 0$

Description : Resets the zero flag Z to 0.

Instruction format :

RZF

0	1	1	1	1	1	1	1	0	0	0	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

0	-	-	-
---	---	---	---

SBC dst, src : Subtract with Carry Flag

✓

Operation : $dst \leftarrow dst - src - C$

Description : Subtracts the contents of src and carry flag C from the contents of dst, then loads the result in dst.

Instruction format :

G format :

SBC.:@:G LEA,#8
SBC.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	1	1	0	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

SBC.:@:A MEM,SEA
SBC.:@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
1	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>; src, by m<5:0>.
When DD = 1, dst and src are exchanged.

Flag status change: Z S V C

Operation size : B W D

*	*	*	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The Z flag is set to 1 when the Z flag = 1 and the operation result = 0; otherwise, cleared to zero.

MSB of sign-extended immediate data must be "0". If immediate data need to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0 ; Byte operation - 00~99H

Word or Double Word operation - 00~79H

A format, DD=0 and SEA=Quick Imm : 0~7

SBCD dst, src : Subtract Decimal with Carry Flag

✓

Operation : $dst \leftarrow dst - src - C$

Description : Decimal-subtracts the contents of src and carry flag C from the contents of dst, then loads the result in dst.

Instruction format :

G format :

SBCD.@@:G LEA, #8
SBCD.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	1	0	1	0	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

SBCD.@@:A MEM, SEA
SBCD.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	1	m3	m2	m1	m0
0	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>

and src is memory specified by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

Operation size : B W D

*	-	-	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The Z flag is set to 1 when the Z flag = 1 and the operation result = 0; otherwise, cleared to zero.

MSB of sign-extended immediate data must be "0". If immediate data need to be sign-extended and its MSB is "1", sign-extended bits are all 1s, and data isn't expressed in BCD.

G format and II=0 ; Byte operation - 00~99H

Word or Double Word operation - 00~79H

A format, DD=0 and SEA=Quick Imm : 0~7

SCF : Set Carry Flag

Operation : $C \leftarrow 1$

Description : Sets the carry flag C to 1.

Instruction format :

SCF

0	1	1	1	1	1	1	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

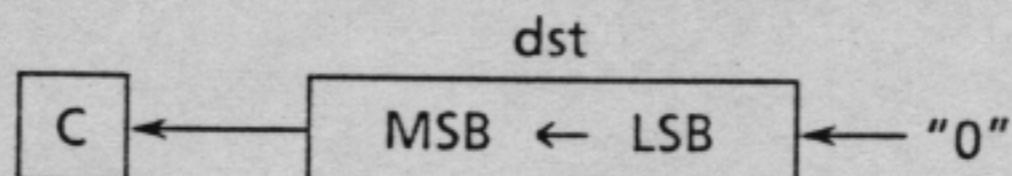
-	-	-	1
---	---	---	---

SLA dst, num : Arithmetic Shift Left

Operation : $C \leftarrow dst < MSB >$, $dst \leftarrow$ shift left dst ,
 $dst < LSB > \leftarrow 0$, repeats by the number of num

Description : Loads the contents of the MSB of dst in carry flag C, shifts the contents of dst to the left, then loads 0 in the LSB of dst. Repeats this by the number of num. The number of the shift in S format is once only.

Description figure :



Instruction format :

S format :

SLA.:@:S LEA,1

S1	S0	1	0	0	1	1	0	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

$M < 7:0 > = dst$

G format :

SLA.:@:G LEA,#8
SLA.:@:G LEA,LEA

S1	S0	0	0		1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	1	0	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

$M < 7:0 > = num$ When $|| = 0$, immediate data (8 bits, signed)
When $|| = 1$, long addressing mode

$N < 7:0 > = dst$, long addressing mode only

A format :

SLA.:@:A MEM,SEA
SLA.:@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	1	m3	m2	m1	m0
0	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When $DD = 0$, dst is memory specified by $A < 12:0 >$
and num is number specified by $m < 5:0 >$.
When $DD = 1$, dst and num are exchanged.

Flag status change: z s v c Operation size : b w d

*	*	0	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format only when $DD = 0$.

When the number of bits to be shifted is 0, dst and the C flag do not change.

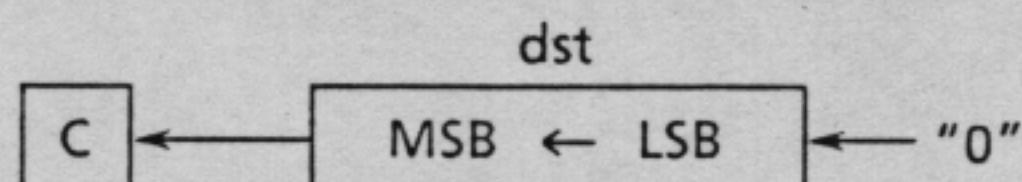
The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4 bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

SLL dst, num : Logical Shift Left

Operation : $C \leftarrow dst < MSB >$, $dst \leftarrow$ shift left dst ,
 $dst < LSB > \leftarrow 0$, repeats by the number of num

Description : Loads the contents of the MSB of dst in carry flag C, shifts the contents of dst to the left, and loads 0 in the LSB of dst. Repeats this by the number of num. The number of the shift in S format is once only.

Description figure :



Instruction format :

S format :

SLL.:@:S LEA,1

S1	S0	1	0	0	1	0	0	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

$M < 7:0 > = dst$

G format :

SLL.:@:G LEA,#8
SLL.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	1	0	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

$M < 7:0 > = num$ When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode

$N < 7:0 > = dst$, long addressing mode only

A format :

SLL.:@:A MEM, SEA
SLL.:@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	1	m3	m2	m1	m0
0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A < 12:0 >$ and num is number specified by $m < 5:0 >$.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D

*	*	0	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

When the number of bits to be shifted is 0, dst and the C flag do not change.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

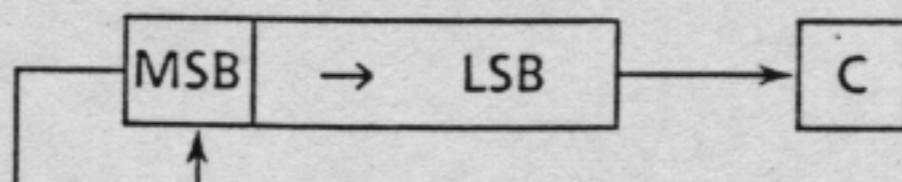
SRA dst, num : Arithmetic Shift Right

J

Operation : $C \leftarrow dst < LSB>$, $dst \leftarrow$ shift right dst , $dst < MSB >$ is unchanged, repeats by the number of num

Description : Loads the contents of the LSB of dst in carry flag C, then shifts the contents of dst to the right (MSB is unchanged). Repeats this operation by the number of num. The number of the shift in S format is once only.

Description figure :



Instruction format :

S format :

SRA.@:S LEA,1

S1	S0	1	0	0	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

M<7:0> = dst

G format :

SRA.@:G LEA,#8
SRA.@:G LEA,LEA

S1	S0	0	0	11	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	1	1	0	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When 11 = 0, immediate data (8 bits, signed)
When 11 = 1, long addressing mode

N<7:0> = dst, long addressing mode only

A format :

SRA.@:A MEM,SEA
SRA.@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	0	1	1	m3	m2	m1	m0
0	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>

and num is number specified by m<5:0>.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c

Operation size : B W D

*	*	0	*
---	---	---	---

O	O	O
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

When the number of bits to be shifted is 0, dst and the C flag do not change.

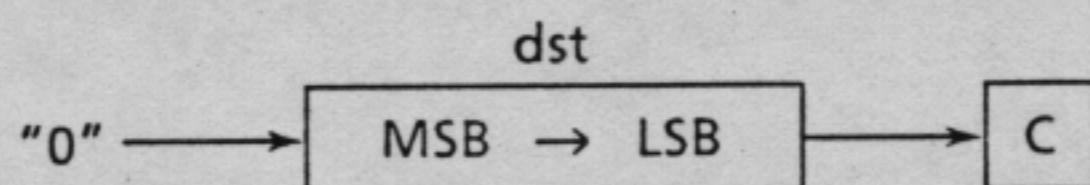
The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

SRL dst, num : Logical right Shift

Operation : $C \leftarrow dst < LSB >$, $dst \leftarrow$ shift right dst ,
 $dst < MSB > \leftarrow 0$, repeats by the number of num

Description : Loads the contents of the LSB of dst in carry flag C, shifts the contents of dst to the right, and loads 0 in the MSB of dst. Repeats this by the number of num. The number of the shift in S format is once only.

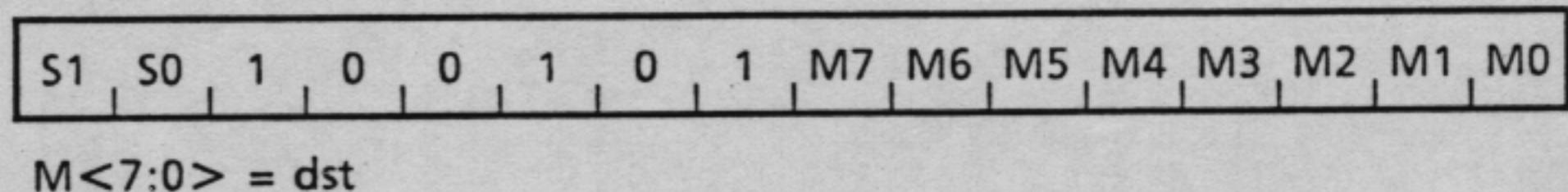
Description figure :



Instruction format :

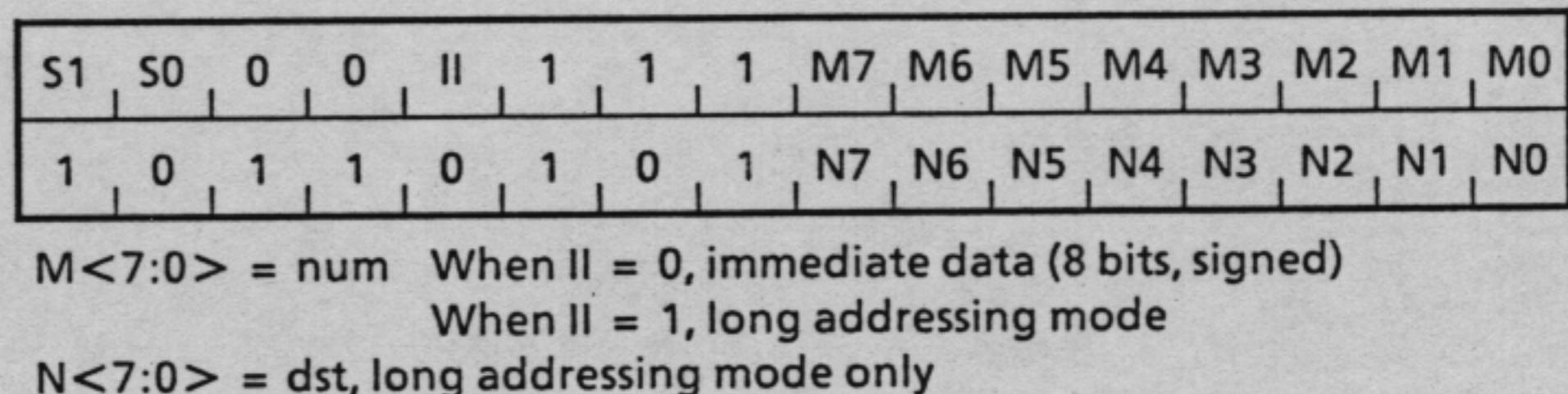
S format :

SRL.:@:S LEA,1



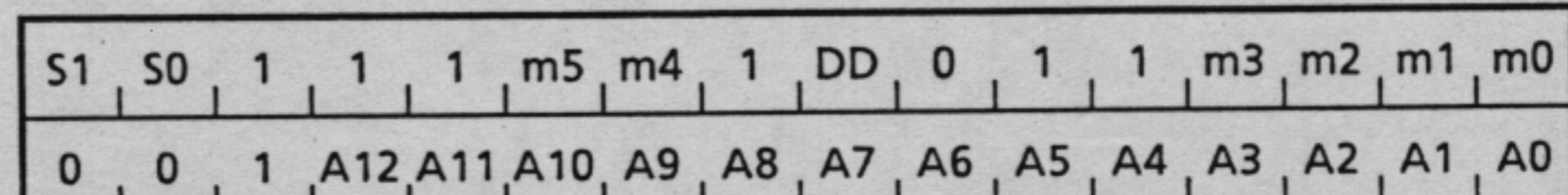
G format :

SRL.:@:G LEA,#8
 SRL.:@:G LEA,LEA



A format :

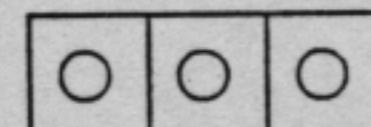
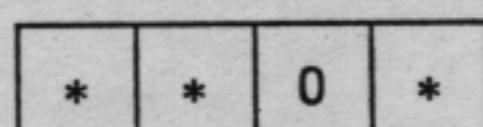
SRL.:@:A MEM,SEA
 SRL.:@:A SEA,MEM



When DD = 0, dst is memory specified by A<12:0>
 and num is number specified by m<5:0>.
 When DD = 1, dst and num are exchanged.

Flag status change: z s v c

Operation size : B W D



Notes : Immediate mode inhibited as dst addressing mode.

$m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

When the number of bits to be shifted is 0, dst and the C flag do not change.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

SSF : Set Sign Flag

Operation : $S \leftarrow 1$

Description : Sets the sign flag S to 1.

Instruction format :

SSF

0	1	1	1	1	1	1	1	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	1	-	-
---	---	---	---

STCF dst, num : Bit Transfer from Carry Flag

J

Operation : $dst < num > \leftarrow C$

Description : Loads the contents of carry flag C in bit num of dst.

Instruction format :

G format :

STCF.0:G LEA, #8
STCF.0:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	0	1	1	1	N7	N6	N5	N4	N3	N2	N1	N0

$M < 7:0 > = num$ When II = 0, immediate data (8 bits, signed)
 When II = 1, long addressing mode
 $N < 7:0 > = dst$, long addressing mode only

A format :

STCF.0:A MEM, SEA
STCF.0:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
0	1	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A < 12:0 >$
 and num is number specified by $m < 5:0 >$.
 When DD = 1, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D

-	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

 $m5m4 = 11$ (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

SUB dst, src : Subtract

Operation : $dst \leftarrow dst - src$

Description : Subtracts the contents of src from the contents of dst, then loads the result in dst.

Instruction format :

S format :

SUB.:@:S Reg, SEA
SUB.:@:S SEA, Reg

S1	S0	0	R0	0	m5	m4	DD	R3	R2	R1	1	m3	m2	m1	m0
----	----	---	----	---	----	----	----	----	----	----	---	----	----	----	----

When DD = 0, dst is specified by R<3:0> and src by m<5:0>.
When DD = 1, dst and src are exchanged.

G format :

SUB.:@:G LEA,#8
SUB.:@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	0	0	0	0	1	0	1	N7	N6	N5	N4	N3	N2	N1	NO

M<7:0> = src When II = 0, immediate data (8 bits, signed)
 When II = 1, long addressing mode
N<7:0> = dst, long addressing mode only

I format:

SUB.:@:I LEA,#16
SUB.:@:I LEA,#32

0	0	0	0	SS	0	0	1	M7	M6	M5	M4	M3	M2	M1	M0
								#<15:0>							
									#<31:16>						

M<7:0> = dst

A format :

SUB.:@:A MEM, SEA
SUB.:@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	0	0	0	m3	m2	m1	m0
0	0	1	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0> and src by m<5:0>.
When DD = 1, dst and src are exchanged.

Flag status change: z s v c

*	*	*	*
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.

m5m4 = 11 (quick immediate) and m5m4 = 00 (register direct) in S format
allowed only when DD = 0.

Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

SUB 3 dst, src1, src2 : Subtract Trinomial

Operation : $dst \leftarrow src1 - src2$

Description : Subtracts the contents of src2 from the contents of src1, then loads the result in dst.

Instruction format :

SUB3.@ Reg,LEA,#8
SUB3.@ Reg,LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
0	R3	R2	R1	R0	0	0	1	N7	N6	N5	N4	N3	N2	N1	N0

$M<7:0> = src2$ When II = 0, immediate data (8 bits, signed)
When II = 1, long addressing mode

$N<7:0> = src1$, long addressing mode only

$R<3:0> = dst$

Flag status change: z s v c

Operation size : B W D

*	*	*	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as src1 addressing mode.

SVF : Set Overflow Flag

Operation : $V \leftarrow 1$

Description : Sets the overflow flag V to 1.

Instruction format :

SVF

0	1	1	1	1	1	1	1	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	1	-
---	---	---	---

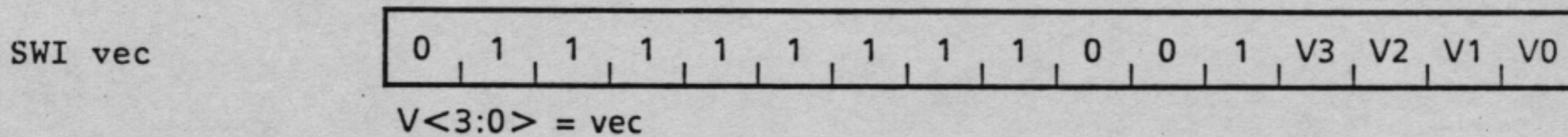
SWI vec : Software Interrupt

Operation : Software interrupt specified by vec.

Description : See interrupt exceptions.

Remarks : The SWI15 software interrupt can only be used by programs like emulators and debuggers using special stack pointers. Cannot be used by general-purpose user programs. For general-purpose user programs, use SWI0 to 14.

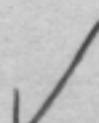
Instruction format :



Flag status change: z s v c

-	-	-	-
---	---	---	---

✓

SZF : Set Zero Flag

Operation : $Z \leftarrow 1$

Description : Sets the zero flag Z to 1.

Instruction format :

SZF

0	1	1	1	1	1	1	1	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

1	-	-	-
---	---	---	---

TJP dst : Table Jump

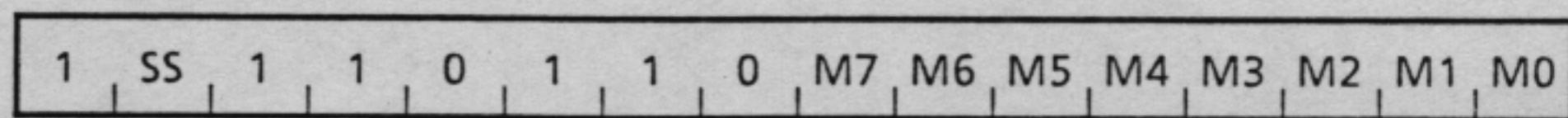
✓

Operation : $PC \leftarrow PC + dst$

Description : Makes a PC relative jump. Adds the contents of dst to the program counter (PC), then generates the jump destination address.

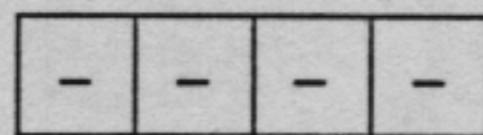
Instruction format :

TJP. @ LEA

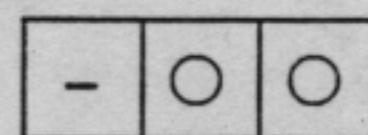


$M<7:0> = dst$

Flag status change: z s v c



Operation size : B W D



Notes : LEA specifies the address where the distance to the jump destination is stored, rather than the jump destination address itself.

Immediate mode inhibited as dst addressing mode.

(RWn++), (--RWn), (RDn++), and (--RDn) in Addressing mode are incremented or decremented by operation size.

TSET dst, num : Test and Set Bit

Operation : $Z \leftarrow \text{inverted value of } dst < num >, dst < num > \leftarrow 1$

Description : Transfers the inverted value of the bit num of dst to the Z flag. Then sets the bit num of dst to "1".

Instruction format :

G format :

TSET.@:G LEA, #8
TSET.@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	1	1	0	1	0	0	N7	N6	N5	N4	N3	N2	N1	N0

$M < 7:0 > = num$ When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

$N < 7:0 > = dst$, long addressing mode only

A format :

TSET.@:A MEM, SEA
TSET.@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	1	1	m3	m2	m1	m0
0	0	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by $A < 12:0 >$ and num by $m < 5:0 >$.

When DD = 1, dst and num are exchanged.

Flag status change: z s v c Operation size : B W D

*	-	-	-
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode. $m5m4=11$ (quick immediate) in A format allowed only when DD=0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

Bus is locked in the read-modify-write cycle.

TST dst : Test Operand

✓

Operation : $CC \leftarrow dst$ test result

Description : Compares the operand with 0 ($dst - 0$). Flags are set according to the test result.

Instruction format :

TST. @ LEA

S1	S0	1	0	1	0	1	0	M7	M6	M5	M4	M3	M2	M1	M0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

$M<7:0> = dst$

Flag status change: z s v c

*	*	0	0
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

UNLK : Unlink Stack Frame

Operation : $SP \leftarrow R2, R2 \leftarrow (SP +)$

Description : Loads the contents of register R2 in the stack pointer (SP), then pops from the stack area the value to update R2.

Instruction format :

UNLK

0	1	1	1	1	1	1	1	1	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Flag status change: z s v c

-	-	-	-
---	---	---	---

XOR dst, src : Exclusive OR

V

Operation : $dst \leftarrow dst \text{ XOR } src$

Description : Exclusive ORs the contents of dst and src, then loads the result in dst.

Instruction format :

S format :

XOR.@:S Reg1,Reg2

S1	S0	1	1	1	1	0	0	T3	T2	T1	T0	R3	R2	R1	R0
----	----	---	---	---	---	---	---	----	----	----	----	----	----	----	----

T<3:0> = dst

R<3:0> = src

G format :

XOR.@:G LEA,#8
XOR.@:G LEA,LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	0	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = src When II = 0, immediate data (8 bits, signed)

When II = 1, long addressing mode

N<7:0> = dst, long addressing mode only

I format:

XOR.@:I LEA,#16
XOR.@:I LEA,#32

0	0	0	0	SS	1	1	0	M7	M6	M5	M4	M3	M2	M1	M0
#<15:0>															
#<31:16>															

M<7:0> = dst

A format :

XOR.@:A MEM,SEA
XOR.@:A SEA,MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
0	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, dst is memory specified by A<12:0>; src, by m<5:0>.

When DD = 1, dst and src are exchanged.

Flag status change: z s v c

*	*	0	0
---	---	---	---

Operation size : B W D

○	○	○
---	---	---

Notes : #<31:16> in I format is used when the operation size is double word.

Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

XORCF src, num : Bit Exclusive OR with Carry Flag

V

Operation : $C \leftarrow C \text{ XOR } \text{src} < \text{num} >$

Description : Exclusive ORs the contents of carry flag C with bit num of src, then loads the result in the carry flag.

Instruction format :

G format :

XORCF.@@:G LEA, #8
XORCF.@@:G LEA, LEA

S1	S0	0	0	II	1	1	1	M7	M6	M5	M4	M3	M2	M1	M0
1	1	0	0	1	1	1	0	N7	N6	N5	N4	N3	N2	N1	N0

M<7:0> = num When II = 0, immediate data (8 bits, signed)
 When II = 1, long addressing mode
 N<7:0> = src, long addressing mode only

A format :

XORCF.@@:A MEM, SEA
XORCF.@@:A SEA, MEM

S1	S0	1	1	1	m5	m4	1	DD	1	0	0	m3	m2	m1	m0
1	1	0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

When DD = 0, src is memory specified by A<12:0> and num is number specified by m<5:0>.
 When DD = 1, num and src are exchanged.

Flag status change: z s v c Operation size : B W D

-	-	-	*
---	---	---	---

○	○	○
---	---	---

Notes : Immediate mode inhibited as dst addressing mode.

m5m4 = 11 (quick immediate) in A format allowed only when DD = 0.

The lower 3 bits of num in G or A format are effective when the operation size is byte ; the lower 4bits, when the operation size is word ; the lower 5 bits, when the operation size is double word.

Instruction execution time (Minimum)

(Number of clock cycles)

Instructions	Format /Condition	Operation size		
		Byte	Word	Double word
ABCD dst, src	G	3	3	5
	A	6	6	10
ADC dst, src	G	2	2	3
	A	5	5	8
ADD dst, src	S	1	1	2
	G	2	2	3
	I	2	2	3
	A	5	5	8
ADD3 dst, src1, src2		2	2	3
AND dst, src	S	1	1	2
	G	2	2	3
	I	2	2	3
	A	5	5	8
ANDCF src, num	G	4	4	5
	A	6	6	9
BCHG dst, num	S	1	1	-
	G	3	3	4
	A	5	5	8
BFEX dst, src, num1, num2		5 + [num1/4 + 1]	3 + [num1/4 + 1]	3 + [num1/4 + 1]
BFEXS dst, src, num1, num2		7 + [num1/4 + 1]	5 + [num1/4 + 1]	5 + [num1/4 + 1]
BFIN dst, src, num1, num2		7 + 2*[num1/4 + 1]	6 + 2*[num1/4 + 1]	7 + 2*[num1/4 + 1]
BRES dst, num	S	1	1	-
	G	3	3	4
	A	5	5	8
BS0B dst, src	G	2	2	9
	A	5	5	14
BS0F dst, src	G	2	2	9
	A	5	5	14
BS1B dst, src	G	2	2	9
	A	5	5	14
BS1F dst, src	G	2	2	9
	A	5	5	14
BSET dst, num	S	1	1	-
	G	3	3	4
	A	5	5	8
BTST src, num	S	1	1	-
	G	3	3	4
	A	5	5	8
CBCD dst, src	G	3	3	5
	A	6	6	10
CHK mem, src	G	-	9	14
	A	-	9	14
CHKS mem, src	G	-	9	14
	A	-	9	14
CLR dst		1	1	2
CP dst, src	S	1	1	2
	G	2	2	3
	I	2	2	3
	A	5	5	8

Instructions	Format /Condition	Operation size		
		Byte	Word	Double word
CPC dst, src	G A	2 5	2 5	3 8
CPL dst		1	1	2
CPSN dst, src, cnts		6	6	7
CPSZ dst, src, cnts		6	6	7
DIV dst, src	G A	15 16	23 24	— —
DIVS dst, src	G A	15 16	23 24	— —
DJNZ dst, disp	dst == 0 dst ≠ 0	— —	6 9	7 10
DJNZC dst, cond, disp	True False dst == 0 False dst ≠ 0	— — —	4 6 9	4 7 10
EX dst1, dst2	S G A	2 3 8	2 3 8	4 5 14
EXTS dst		—	2	2
EXTZ dst		—	2	2
JP dst		—	5	5
LD dst, src	S G I A	1 2 2 5	1 2 2 5	2 3 3 8
LDA dst, src		—	3	4
LDCF src, num	G A	4 6	4 6	5 9
LDS dst, src, cnts		6	6	7
MAC dst, src1, src2		8	9	—
MACS dst, src1, src2		7	8	—
MAX dst, src	G A	5 8	5 8	7 12
MAXS dst, src	G A	5 8	5 8	7 12
MIN dst, src	G A	5 8	5 8	7 12
MINS dst, src	G A	5 8	5 8	7 12
MIRR dst		1	1	3
MUL dst, src	G A	7 10	8 11	— —
MULS dst, src	G A	6 9	7 10	— —
NEG dst		1	1	2
OR dst, src	S G I A	1 2 2 5	1 2 2 5	2 3 3 8

Instructions	Format /Condition	Operation size		
		Byte	Word	Double word
ORCF src, num	G	4	4	5
	A	6	6	9
POP dst		5	5	8
PUSH src		5	5	7
PUSHA dst		-	6	9
RL dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
RLC dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
RLM dst1, dst2, num		7 + [n/4]	7 + [n/4]	8 + [n/4]
RR dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
RRC dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
RRM dst1, dst2, num		7 + [n/4]	7 + [n/4]	8 + [n/4]
RVBY dst		1	1	3
SBC dst, src	G	2	2	3
	A	5	5	8
SBCD dst, src	G	3	3	5
	A	6	6	10
SLA dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
SLL dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
SRA dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
SRL dst, num	S	1	1	2
	G	3 + [n/4]	3 + [n/4]	3 + [n/4]
	A	5 + [n/4]	5 + [n/4]	7 + [n/4]
STCF dst, num	G	4	4	5
	A	6	6	9
SUB dst, src	S	1	1	2
	G	2	2	3
	I	2	2	3
	A	5	5	8
SUB3 dst, src1, src2		2	2	3
TJP dst		-	5	6
TSET dst, num	G	4	4	6
	A	7	7	11
TST dst		1	1	2
XOR dst, src	S	1	1	2
	G	2	2	3
	I	2	2	3
	A	5	5	8

Instructions	Format /Condition	Operation size		
		Byte	Word	Double word
XORCF src, num	G A	4 6	4 6	5 9

Notes : [num1/4 + 1] is a value obtained by dividing the number of shifted bits by 4 and rounding up to an integer.

[n/4] is a value obtained by dividing the number of shifted bits by 4 and rounding up to an integer.

(Number of clock cycles)

Instructions	Condition	
JRC cond, disp	True	6
	False	3
JRBC num, abs, disp	True	9
	False	8
JRBS num, abs, disp	True	9
	False	8

(Number of clock cycles)

Instructions	SP	
	16BIT	32BIT
CALL dst	11	12
CALR disp	10	11
LINK disp	5	6
RET	8	9
RETD disp	8	9
UNLK	5	9

(Number of clock cycles)

Instructions	Banks	Memory stacks
RETI	12	17
RETS	12	17
SWI vec	18	25

(Number of clock cycles)

Instructions	
CCF	1
CSF	1
CVF	1
CZF	1
DI	2
EI	2
HALT	3
JR disp	5
NOP	1
RCF	1
RSF	1
RVF	1
RZF	1
SCF	1
SSF	1
SVF	1
SZF	1